Five Multiscan Monitors: A Pixel-for-Pixel Comparison

Amazing COMPUTING™

Your Original AMIGA™ Monthly Resource

Vol. 3 No. 10 October 1988
US $3.95 Canada $4.95

Amiga Your BASIC Tool

Create Graphic Objects in AmigaBASIC

DSM MC68000 Disassembler Review

Investigating Fast Fourier Transforms

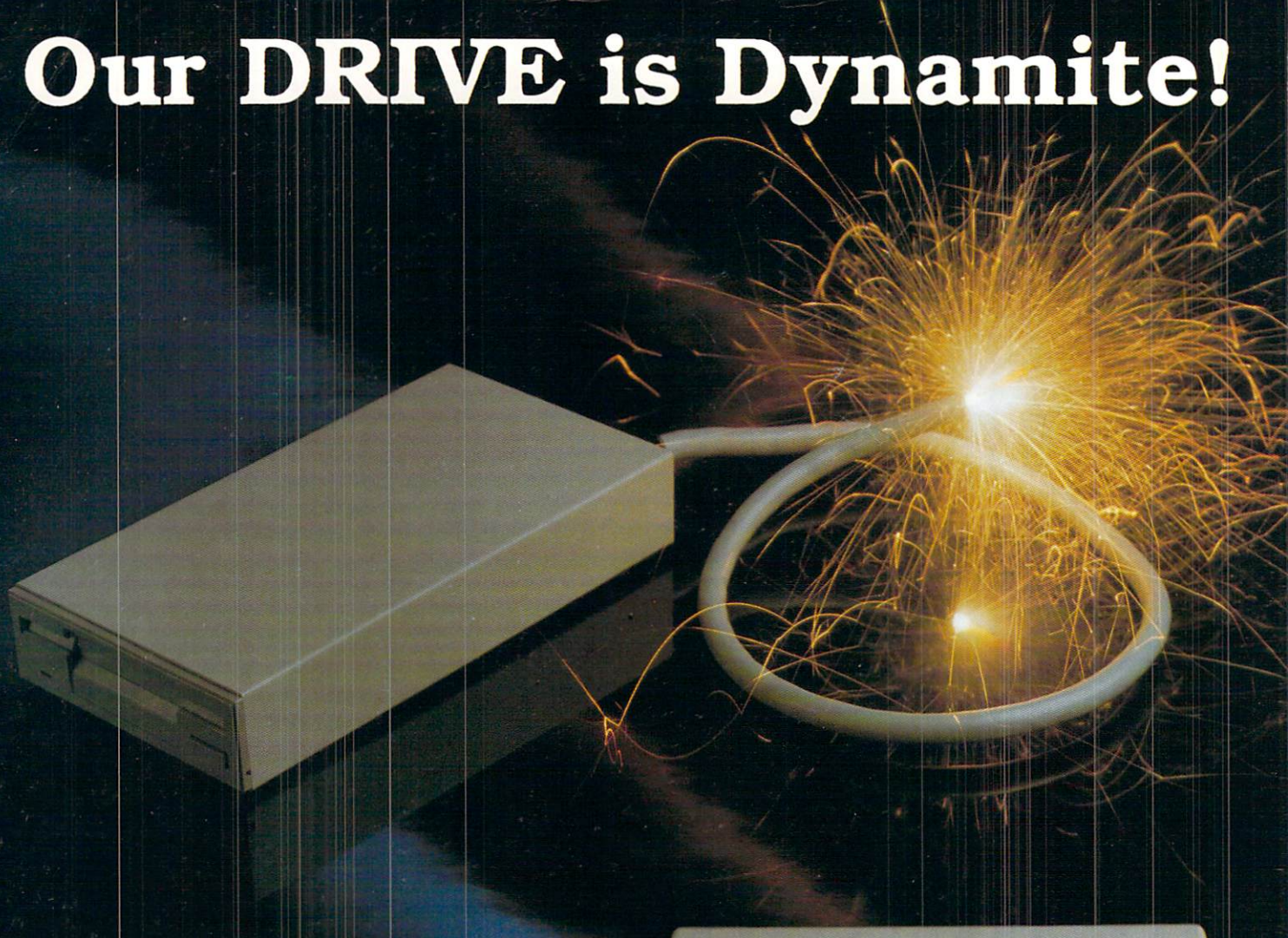
A First Look at Deluxe PhotoLab

Also

SuperBase Professional Tutorial:
Record Keeping for Free-Lancers

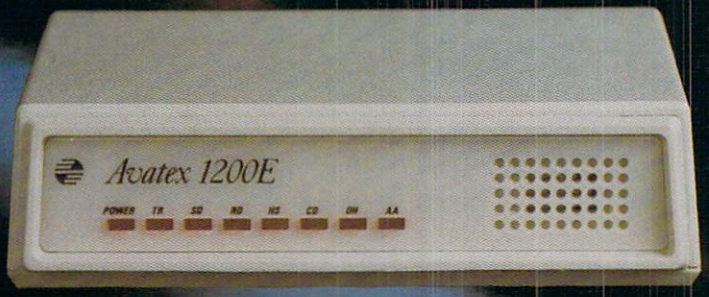
10
0 74470 74710 9

Our DRIVE is Dynamite!



The AIR DRIVE

Dual-Sided, 880K
Color-coordinated chassis and faceplate
Smaller and lighter than the Amiga drive
External drive pass-through
Extra long 30" interface cable
Spring-loaded dust door
Compatible with all Amiga systems
One year warranty
Internal drive also available
CALL FOR DEALER NEAREST YOU.



$79 Small and powerful. The Avatex 1200E.

Avatex MODEMS

$105	1200P
$99	1200HC
$169	2400
$69	1200i
$159	2400i

FREE WITH EACH MODEM
Amiga communication software
& CompuServe access time.

MEGATRONICS

To Order: 800-232-6342

Ask about our **LOWEST PRICE GUARANTEE**

INSIDE UTAH: (801) 752-2642
FAX: (801) 752-8752
CALL FOR A FREE CATALOG.
CREDIT CARDS VERIFIED FOR YOUR PROTECTION.

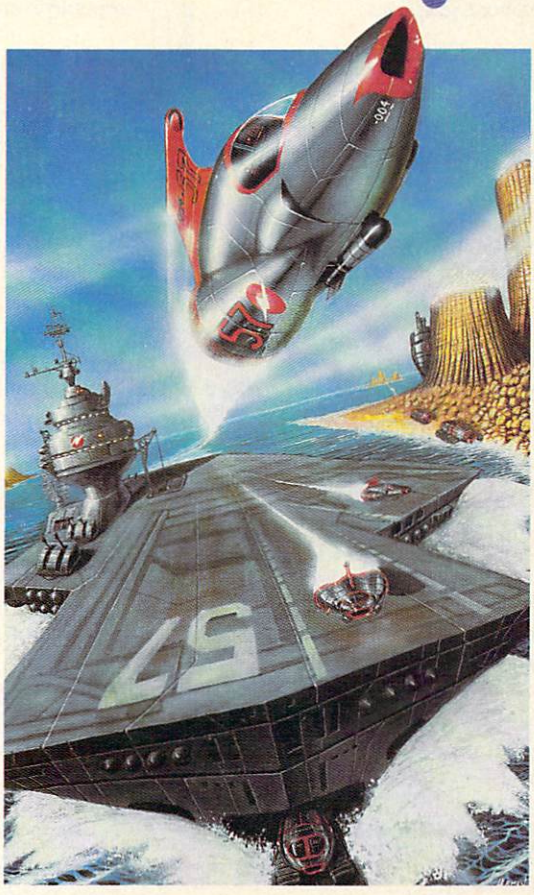
MEGATRONICS, INC. BOX 3660, LOGAN, UT 84321

CARRIER COMMAND

ENCOUNTER HIGH SPEED ACTION IN AN INTELLIGENT GAME OF TACTICAL WARFARE.

THE TIME: The 22nd Century.
THE PLACE: 64 islands in the Southern Ocean.
THE PROBLEM: A worldwide energy crisis.
THE MISSION: Gain control of the islands, set up centers to mine, recycle and produce materials to form a large network of power plants.
JUST ONE MORE PROBLEM: Enemy terrorists are destroying the islands. . . one by one.

FAST-PACED ARCADE ACTION: You are in control of up to 4 aircraft and 4 amphibious vehicles simultaneously. Capture enemy islands and destroy its forces.
STRATEGIC TACTICAL WARFARE: Conduct war maneuvers in a huge territory that includes over 60 islands. Protect your ship with defense drones and 360-degree turret mounted laser cannon with telephoto tracking.
SENSATIONAL GRAPHICS: Three-dimensional solid filled graphics, smooth scrolling, fabulous sound and special effects.
ADDICTIVE, HOURS OF PLAY: Your choice of action game or strategy game, plus save-game option provides hours and hours of extraordinary adventure!



REALTIME GAMES SOFTWARE LIMITED

RAINBIRD

P.O. Box 2227,
Menlo Park, CA 94026,
415/322-0900

HOW TO ORDER: Visit your software dealer today, or call (800) 227-6900 from U.S. or Canada for Visa, MasterCard, Amex or C.O.D. To order by mail, send check/money order: Rainbird, P.O. Box 8123, San Francisco, CA 94128. CA add 6% sales tax and TX add 7½%. Shipping/handling is $4.50. 2-3 weeks for delivery.

AVAILABLE: IBM CGA/EGA/VGA $39.95. Amiga and Atari 512K $44.95. Macintosh 512K $49.95. Commodore 64/128 coming soon.

Amazing COMPUTING™

Volume 3, Number 10

CONTENTS

Amazing Features

Amazing Columns

Amazing Reviews

Amazing Departments

Amazing Dealers

The following are Amazing Dealers, dedicated to supporting the Commodore-Amiga™. They carry Amazing Computing™, your resource for information on the Amiga™. If you are not an Amazing Dealer, but would like to become one, call PiM Publications, Inc.:

1-508-678-4200

Alaska
That Computer Place — Anchorage
Juneau Electronics — Juneau

Alabama
Abax Data Systems — Huntsville
Alabama Book Store — Tuscaloosa
Command Computers — Birmingham
Madison Book & Computer — Madison
Mel's Photo & Computer Shop — Montgomery
The Computer Nut — Birmingham
The Computer Image — Birmingham
Universal Computer System — Mobile

Arkansas
Commodore Connection — Sherwood
Microtronix — Fort Smith
The Micro Shop — Little Rock

Arizona
Books Etc. — Tempe
Books Brothers Ltd. — Tucson
Computer Library, Inc. — Tempe
Computer Works — Glendale
Copperstate Business Systems — Phoenix
North American Digital — Tucson
Try Tech Inc. — Pheonix

California
3 E Software And Systems — Hayward
Aday International — Foster City
American Mags & Books Ltd. — Englewood
Augnet — Fremont
Balboa Office Supply — San Diego
Bits & Bytes — Fairfield
Bits n Pieces Comp Prod. — Anaheim
Bookland — Marysville
Brown Knows Computing — Redlands
CAL-Tech Bookstore — Pasadena
Candy Computer — Elk Grove
Centerfold News — Los Angeles
Central Park Bookstore — San Mateo
Century Computer Systems — La Habra
Circus of Books — Los Angeles
Circus of Books — Los Angeles
Commodore Land — Alhambra
Commotech — National City
Computer Attic — Palo Alto
Computer Literacy — Santa Clara
Computer Literacy Books — Sunnyvale
Computer Nook, Inc. — San Bernadino
Computer Outlet — San Diego
Computer Time — Citrus Hts
Coronet Newsstand — Oceanside
Covina Bookstore — Covina
Dataphile — Watsonville
Delta Spectrum — Laguna Beach
Delaurer's News Agency — Oakland
Discount Computer Station Inc. — Long Beach
Empire Computers — Santa Rosa
Encinitas News — Encinitas
ECX Computer Co. — Walnut Creek
Fisher Computers + Software — Rancho Cucamonga
Harding Way News — Stockton
Heinrichson, Schneider & Young, Inc. — Berkeley
Home Computing Center — Concord
Home Computing Center — Mill Valley
Home Computing Center — San Leandro
Home Computing Center — San Bruno
HT Electronics — Sunnyvale
International Telecine — Santa Ana
K J Computers — Granada Hills
Little Professor Books — Placentia
LTC Computers — Santa Ana
Lyon Enterprises — Berkeley
Micro Flight — San Jacinto
NewsBest — Davis
Northridge Newsstand — Northridge
Op Amp Tech Books — Los Angeles
Paras Newsstand — San Diego
Pat's Electronics — North Highlands
R&R Software — Chula Vista
Ridgecrest Computer Ctr. Inc. — Ridgecrest
Sawyer's News Inc. — Santa Rosa
S.O.S. Computers — Los Angeles
Software First — Santa Rosa
Software Cellar — Anahiem
Software Central — Pasadena
SoftwareLand — San Diego
Software Plus — Citrus Heights
Software Solutions Plus — El Cerito
Stacy's #5 — Modesto
Technitronics Service — South Pasadena
The Booksmith — San Francisco
Third Avenue News — Chula Vista
Tower Books — Chico
Tower Books — Citrus Hts.
Tower Books — Concord
Tower Books — Mountain View
Tower Books — Sacramento
Tower Books — San Mateo
The Computer Room — Scotts Valley
Universal News — Hollywood
WCI Computer Central — Eureka
Warner Engineering — San Diego
Winner's Circle — Berkley
World Book & News — Hoolywood

Colorado
Aurora Newsstand — Aurora
Book Tech Distributing — Englewood
Citadel Computers — Colorado Springs
Computer Discount — Denver
Computer Reports — Boulder
Computer Room — Aurora
Micro Computer World — Fort Collins
Micro World — Lakewood
Sun Country Computers — Colorado Springs

Connecticut
Aetna Life Club Store — Hartford
Amigo Business Computer — Norwalk
Computer Outlet — Pa'Vernon
Computer & Software Connections — Plantsville
Demic Electronics — Hamden
Friendly Computers — Plantsville
Mnemonics Inc. — Norwalk
Personal Computer Center — Norwich
Software City — Orange
Software Kingdom — East Hartford
Tricom Computers — North Windham

District of Columbia
Programa — Washington

Delaware
Castle Video — Newark
Newark Newsstand — Newark

Florida
A A Computers — Jacksonville
Chuck's Newsstand — Valleach
City Newsstand — Melbourne
Commodore Country Inc. — Pinellas Park
Computer Bar — Pennsacola
Computer Base — Brandon
Computer Image — Miami
Computer Terminal Inc. — Sebring
Computers + Video Creations — Palm Beach
Computers 4 Rent — W. Palm Beach
Computers Plus — S. Daytona
Computer Room — Ft. Lauderdale
Cotronics Inc. — Stewart
Creative Equipment — Miami
Education Computers Etc. — Talahasse
Family Computers — Sarasota
Florida Book Store — Gainesville
Gulf Coast Computer Exchange — Panama City
Heartland Computers — Daytona Beach
Joel's News, Inc. — Miami
Micro Computer Center — Largo
New Age Electronics — St. Petersburg
Newton's Video — Boca Raton
PC Collection — Coral Gabels
Program Store — Clearwater
Prospect News & Books — Ft. Lauderdale
Random Access Computers Inc. — Ft. Walton Beach
RGB Computers and Video Creations — Palm Beach Gardens
Sunny's At Sunrise, Inc. — Sunrise
Computer Room — Ft. Lauderdale
The Open Door — Cocoa

Georgia
BFJ Enterprises — Roswell
Cardinal Computers — Dalton
Software South Inc. — Savannah
The 64 Store — Atlanta

Hawaii
Astro Computers & Electronics — Aiea
Computer House Inc. — Honolulu
Computerland — Honolulu
Data-One Computer Center — Honolulu
J & C Repair — Aiea

Idaho
A.B.I. Computer & Video — Nampa
Boise Computer & Video Inc. — Boise

Illinois
A Bitcketer — Rockford
Audio Visual Req. — Chicago
Computer Software Connection — Moline
DigitalWorld — Addison
DOS Computer Center — Niles
I.C.Y. Data Center Inc. — Mundelein
Keep In Pace Computers — Champaign
M2J2 Inc. — Champaign
Kroch's & Brentano's — Chicago
Kroch's & Brentano's — Schaumburg
Kroch's & Brentano's — Orland Park
Kroch's & Brentano's — Aurora
Micro Age — Oaklawn
Micro Ed Enterprises Inc. — Chicago
Ring Software — Geneva
Software City — Belleville
Software Or Systems — Fairview Hts.
Software Plus — Wheeling
Software Plus Chicago — Chicago
Software Plus West — Hanover Park
The Memory Expansion — Bloomington
Unique Computer — Sterling
Warren Radio/FRANCO — Peoria

Indiana
Bookland — Indianapolis
Burkat Computer Ctr. — South Bend
Bytrex of Angola — Angola
Computer Corner — Marion
Computer People Inc. — Michigan City
Computer Products Unlimited — Ft. Wayne
CPU Inc. — Indianapolis
Indiana News — Indianapolis
Micro Computer Centers Inc. — Indianapolis
Reese Equipment Co. — Plymouth
The Book Corner — Bloomington
The Computer Room of Merrtville — Merrtville
Vons Computers — West Lafayette

Iowa
Cronet Computers — Des Moine
Micro Computer Applications LTD — Marshalltown
Soft Search — Ames

Kansas
Arc Electronic Distributors — Wichita
J & L Electronics — Liberal
MidKansas Computers — Newton
Midwest Computers — Manhattan
Software City — Overland Park

Kentucky
Computer Dimensions Inc. — Lexington
Computer Mania Inc. — Florence
Computer Store of Kentucky — Louisville
Hawley Cooke — Louisville
Reality World Computers — Lexington

Louisiana
City Newsstand — Baton Rouge
Computer Time — Shreveport
Lakeside News — Metairie
Modern Business Machines — Metairie
Software Center International — Metairie
Software Source — Lafayette
The Book Shelf — Slidell
The Computer Clinic — Lafayette

Maine
Computer Barn — Damariscotta
Hands On Computing — Westbrook
Radio Shack Associate Stores — Jay
Skylight Software — Belfast

Maryland
Buried Treasure — Rockville
Cal Com Inc. — Silver Spring
Compuvision Vision Center — Columbia
Greetings and Readings — Towson
Knowledgeware Systems — Kensington
Micro Computer Center — Baltimore
Professional Micro Services — Baltimore
Software Advantage — Rockville
Software N Things — College Park
Waldorf Computer — Waldorf

Massachusetts
Computer Zone — North Attleboro
General Computer Store — Framingham
HCS Computer Center — Marshfield
LCA Video & Computer Ctr. — Norwood
Memory Location — Wellsley
OmniTek Computers — Tewksbury
Pioneer Valley Data Equipment — Amherst
Softpro — Burlington
The Bit Bucket — West Newton
The Software Haus — Quincy
Tycom Inc. — Pittsfield

Michigan
Basic Computer Center — Grand Rapids
CompuSoft — Kalamazoo
Computer Emporium — Lansing
Direct Access — Lansing
Central Park Computers — Okemos
Global Computer Center — Dearborn
I/O Software — Flint
Michigan News Agency — Kalamazoo
Michigan Software — Novi
Micro World — Livonia
Micro World — Milford
Micro World — Roseville
Micro World — Southfield
Micro World — Southgate
Micro World — Sterling Heights
Microworld Computer Center — Dearborn
Midland Computer Shop — Midland
Readmore Newscenter — Lansing
Rochester Book Center — Rochester Hills
Roseville Computer Store — Warren
Slip Disk — Madison Heights
Software City — Grand Rapids
Specialists Computer — Livonia
State Street Computers — Ann Arbor
T.J. Enterprises — Clinton
The Edge Connector — Clio
United Computer — Canton
Ye Olde Computer Shoppe — Ypsilanti
Video Asis — Sterling Hts.

Minnesota
Computer Excercise World — Minneapolis
Computer Stuff — Minneapolis
Computers, etc. — Eagan
JMH Software — Maple Grove
Lookahead Software Computer — Northfield
Midwest Computer Services — Clearwater
Shinder's — St. Paul
Shinder's — Maplewood
Shinder's — Burnsville
Shinder's — Minnetonka
Shinder's — Edina
Shinder's — Crystal
Shinder's Read More — Minneapolis
Specialists IN — Hopkins
Valiant Inc. — Stillwater

Mississippi
Computer Connection — South Haven
National Computer Center — Ocean Springs

Missouri
Associated Computer Services — Springfield
Brands Mart Computers — Kansas City
Carthage Electronics — Carthadge
Data Byte Computer Co. — St. Louis
Software City — Independence
Systems Plus — Overland Park
Systems Plus of St. Louis — St. Louis

Montana
Appelgren Computer Systems — Great Falls

Nebraska
Double E Electronics — Omaha
My Com Inc. — Plattsmouth

Nevada
Century 23 Inc. — Las Vegas
Computer World — Las Vegas
Tower Records — Las Vegas

New Hampshire
Dartmouth Bookstore — Hanover
Diversified Computers — Keene

New Jersey
Business System Professional — Scotch Plains
Family Computer Centres — South Orange
Morris County Stationers — Chester
Software Station — Rockaway
Take One More Byte — Hoboken
The Program Store — Eaton Town
The Program Store — Wayne
Unlimited Software Inc. — Rutherford

New Mexico
New Horizon Computer System — Alamogordo
Page One Newsstand — Albuquerque
Road Runner — Espanola
Technological Concepts — Las Cruces

New York
Amuse — New York
Amicus- Hudson Valley — Wappingers Falls
Amigo Business Computer — East North Port
Byte Shop — Merrick
Compurtie Center — White Plains
Computer Outlet — Jamestown
Computers Etc. — Syracuse
Contemporary Computing — Latham
Creative Computers — Johnson City
Cromland Inc. — Brooklyn
Leeder's Video — Schenectady
Leigh's Computers — New York
MCP Associates — Astoria
Mnemonics — North Port
Readers Market — Tonawanda
Software & Such — Ballston Lake
Software City — Forest Hills
Software Link — White Plains
Software Supermarket — Tonawanda
Star Tech Systems — Massena
Synfaire Trading — New York
Synput Computers — Brooklyn
Tadiran — New York
Tedrow Business Products — Rochester
The Computer Cellar — Albany
Unicorn Electronics — Johnson City
Video Computer Bin — Brooklyn
Video Computer Center — Rome
Village Computer Inc. — New York
World Computers Inc. — Hicksville

North Carolina
Digitz — Raleigh
K&S Newsstand — Clemmons
K&S Newsstand — Kernersville
K&S Newsstand — Winston-Salem
News Center — Cary
Parker News Inc. — High Point
Software City — Raleigh
Software City — Charlotte
TDS Computers — Durham
Triad Computers — Winston-Salem

North Dakota
Computer Associates — Fargo
G. F. Computers — Grand Forks
Software House — Fargo
The Computer Store — Minot

Ohio
B5 Software — Columbus
Computer Network — Newark
Computer Site — Cleveland
Earthrise Micro Systems — Columbus
Eddy's Cigar Store — Elyra
Fairborn Home Computer — Dayton
Lakewood International News — Lakewood
Lakes Consumer Electronics — Akron
Leo's Book & Wine Shop — Toledo
Micro Center — Columbus
Microtech Computer Center Inc. — Toledo
Microware Magic — Fairfield
News Shop — Kent
North Coast Programming — Willoughby
Paper & Ink — Tiffin
Software Centre International — North Olmsted
Software and More — Cincinnati
Software Connection — Mayfield Hts.
Specialist Computer System — Toledo

Oklahoma
Ltlle Professor Books — Stillwater
Merit Computers — Oaklahoma City
Micro Link Computer Store — Tulsa
P C Tech — Tulsa
Second Hand Software — Oklahoma City
Video-Comp Inc. — Lawton

Oregon
Andromeda — LaGrande
Clackamas Computers — Beaverton
Clackamas Computers — Clackamas
Comm-Shack — Salem
Computron Business Systems — Portland
PC Express — Eugene
Powell's Tech — Portland
Rich Cigar Store — Portland
Sixth & Washington News — Portland
Software Pipline — Beaverton
Tech Books — Portland

Pennsylvania
Alpha Omega EBE — Warren
ATD Software — Sayre
Basic Computer Systems — Erie
Basic Computer Systems — Hermitage
Circle Computer — Ephrata
Computer Wurx — Stroudsburg
Computers and Games — Reading
Downington Computer Center — Downington
East Coast Software — Hershey
Floppy Disk — Mechanicsburg
Floppy House Software — Palmyra
Gene's Books — King of Prussia
Micro Master Inc. — Monaca
North Central Digital Systems — Danville
Pittsburgh Computer Store — Pittsburgh
Software Hut Inc. — Philadelphia
Tasco Computer Center — Lancaster
Triangle Computers — Indiana
Walden Hill Images — Berlin

Rhode Island
Micro Limits — Smithfield
Micro Limits — Warwick
Software Connection — Warwick

South Carolina
Avcom International Inc. — West Columbia
Micro Computer Depot — Sumter
Watson's Jewlery — Chesterfield
Software Haus — Charleston Hts

South Dakota
Computer Business Center — Rapid City

Tennessee
ComputerLab of Memphis — Memphis
Computer Hardware Services — Nashville
The 64 Store — Bartlett

Texas
All Systems Go Software — San Antonio
Amazing Computer Systems — Carrollton
Bookstop — Houston
Bookstop — San Antonio
Byte Shop — Austin
C.E.M. Corporation — Brazoria
Colonel Video — Houston
Computer Age — Houston
Computer Age — Dallas
Computer Magic — Austin
Computer Skills — Euless
FNS Computer Services — Texarkana
Inter Computing Inc. — Grand Prarie
Maxwell Book Inc. — Desota
Metropolitan Computer — Richardson
Micro Search — Houston
Software Library — Wilchita Falls
Software Terminal — Fort Worth
The Computer Experience — San Antonio
The Computer Stop! — Amarillo

Utah
Magazines and Gifts — Salt Lake City
People's Computers — Provo

Vermont
Ray Supply Inc. — South Burlington

Virginia
Family Computer Center — Fairfax
Family Video & Computer Center — Newport News
I/O Computer — Norfolk
Software Central — Norfolk
Software City — Richmond
Tract Software — Prince William
Virginia Micro Systems — Woodbridge

Washington
Amazing Electronics — Spokane
A.P.P.L.E. CO-OP — Renton
Apple Valley Computers — E. Wenatchee
Bellingham Computer Center — Bellingham
Bits, Bytes and Nibbles — Spokane
Bull Dog News — Seattle
Byte World — Kent
Computer Exchange Services — Bellingham
Computer Quest — Edmonds
Computers + Inc. — Tacoma
Family Computers — Everett
Micro Age Computer Store — Bellevue
Nybbles & Bytes — Tacoma
Omni International — Seattle
Port Book and News — Port Angeles
Programs Plus — Seattle
Real Time Systems — Seattle
Take A Byte — Kennewick
The 68000 Club — Spokane
Tower Books — Bellevue
Y Not Magazines and Boks — Federal Way

Wisconsin
20th Century Books — Madison
Computer Software Center — Milwaukee
Fox Valley Personal Computer — Oshkosh
Madison Programmer's Guide — Madison
Mayfair Computers — Milwaukee
Mom's Computers — Stevens Point
River City Software — La Crosse
Software City — Racine
TMW Software Inc. — Wausau

Wyoming
Byte Emporium — Laramie
Crazy Mountain Electronics — Cooper

Canada
Compute or Play — Alberta
Computerworks — Alberta
Computer Shop — Alberta
Data Pro — Alberta
Computer Shop of Calgary — Alberta
The Software House — Alberta
Hindsin Computing — Alberta
Red Deer Micro Systems — Alberta
Memory Land — Alberta
Compu Care — British Columbia
Computer King — British Columbia
Corsi — British Columbia
Nutec — British Columbia
Pacific Business Technologies — British Columbia
Sprta — British Columbia
Strider — British Columbia
Super Software — British Columbia
Canadian Computer Era — Manitoba
Micromart — Manitoba
ComputerWorld — Manitoba
J.W. Allen — Manitoba
Kobetak — Manitoba
Minicomp — Manitoba
Arkon — Ontario
Chatam Computer — Ontario
Computerland — Ontario
Comspec — Ontario
Electronics 2001 — Ontario
Hastings Data — Ontario
The Computer & You — Ontario
Thornhill — Ontario
Software Co. — Quebec
Detronie Solare — Quebec
Merodata — Quebec
Info Merveor — Quebec
Laval Micro — Quebec
Vision Tronique — Quebec
Distribution Visipro — Quebec

Distributed in Canada by
Ingram Software Distributors — Concord, Ontario
Compulit Distributors — British Columbia
Phase 4 Dist. Inc. — Calgary, Alberta

Argentina
Logic Computer — Buenos Aires

Australia
Computermart — Leederville,WA
Diskworks — Greenwich, NSW
John Verhoeven — Cloverdale, WA
Maxwell Office Equipment — Abbotsford, Melburn
Run Soft — Melbourne

France
A.M.I.E. — Paris

Saudia Arabia
Abdulla Fouad & Sons — Daman

Scotland
The Amiga Center of Scotland — Edinborough

Sweden
Nicholas Peerson — Lund

Amazing Computing™ is also available in most B. Dalton Booksellers, B. Dalton Software Stores, Crown Books, Software Etc., selected WaldenBooks Stores, and Walden's Software Store locations.

A M A Z I N G M A I L

Dear Amazing Computing,

While reading the article called "Amiga Libraries and the FFP and IEE Math Routines" (V3.8 p. 91), I noticed it said that OpenLibrary causes the computer to search the DEVS: directory on the disk. I believe this is wrong. If I am not mistaken, OpenLibrary searches the LIBS; directory, not the DEVS: directory. Other than that minor error, I found the article to be informative and helpful. Also, I really like the Amazing Column "C Notes from the C Group." Keep up the good work on the magazine!

Sincerely,
Greg Menzel
Minnesota

You are right! OpenLibrary() causes the system to search for a library in the system library list. If the library is not RAM resident, OpenLibrary() searches the directory currently assigned to LIBS:

Dear AC:

I wish to praise your magazine for the consistently informative articles constantly coming each month. One regular article especially important to me now is Rich Falconburg's "The Command Line."

As I eagerly wait for each new installment of "The Guide to the CLI," I too have been frustated and exasperated beyond recognition some nights as I fight through some awkward disk maneuvers. The last issue (V3.7, p.70) was great, until the last few paragraphs. Rich, ya did a fine job explaining the procedure for editing the "startup-sequence," but how do I save changes to disk? Remember, I am a Clutz Learning It !

Well, after literally hours of manual searching, here is a short list of CLI commands for editing which other readers might appreciate:

ESC X = exits and saves changes to disk (my favorite);
ESC Q = quits and does NOT save changes to disk;
ESC T = moves cursor to top of file;
ESC B = moves cursor to bottom of file;
CTRL B = deletes whole line cursor [is] currently on;
CTRL A = adds a blank line below the current cursor position.

And, remember to push RETURN.

I hope others will benefit from these necessary and useful CLI disk commands.

Keep the great materials coming. How about more issues on computer music?!

Sincerely,
Robert G Burnet
Canada

Dear Amazing Computing,

As the most informative and knowledgeable computer publication available, I'm asking your assistance in a problem I have with my 2 meg memory expansion board and chassis from Micron Technologies for the A1000. I bought the board while on leave in Wisconsin and hand carried it on the plane back to where I'm stationed, the Republic of Philippines. When I hooked it up to my A1000 and booted with KS and WB 1.2, I found that the system will freeze up within a couple of minutes. I do not get a Guru; everything just stops. No input from any source is accepted. This happens whether I'm in CLI (CTRL-D on boot), or let WB load and use icons. I ran PUmemtest V1.1, and it states the expansion libraries cannot be opened. Memtest V1.1 ran through all the way with no apparent problems. I rebooted after Memtest 1.1 with KS and WB 1.1, and exactly the same thing happened. Total freeze-up of the system. I tried this on two other A1000's and the same thing happened.

I bought all kind of new things for my A1000 while on leave, and I'm anxiously awaiting a fix for this so I can use these new products to their full potential. Thanks for any help.

Sincerely,
Keith K.Fisher, TSgt USAF
California

A quick call to Micron Technologies yielded a very enthusiastic response from Becky Shriver. Her instructions are as follows:

1. Firmly seat the board in the package.
2. Firmly push the connector onto the Expansion Bus.

Amazing COMPUTING

ADMINISTRATION

Publisher:	Joyce Hicks
Assistant Publisher:	Robert J. Hicks
Circulation Manager:	Doris Gamble
Asst. Circulation:	Traci Desmarais
Asst. Circulation:	Donna Viveiros
Corporate Trainer:	Virginia Terry Hicks
Traffic Manager:	Robert Gamble
International Coordinator:	Marie A. Raymond

EDITORIAL

Managing Editor:	Don Hicks
Co-Editor:	Ernest P.Viveiros Jr.
Co-Editor:	Michael T. Cabral
Hardware Editor:	Ernest P. Viveiros Sr.
Music & Sound Editor:	Richard Rae
Copy Editor	Michael Creeden

PRODUCTION

Art Director:	Keith Conforti
Illustrator:	Brian Fox
Production Manager:	Rico Conforti
Associate Prod. Mgr:	Mark Thibault

ADVERTISING SALES

Advertising Manager:	John D. Fastino
Marketing Assistant:	Melissa J. Bernier

1-508-678-4200
FAX 1-508-675-6002

SPECIAL THANKS TO:
Buddy Terrell & Byrd Press
Betsy Piper at Tech Plus
Bob at Riverside Art, Ltd.
Swansea One Hour Photo
Boston Jewelry & Loan of Fall River

Amazing Computing™ (**ISSN 0886-9480**) is published monthly by PiM Publications, Inc., Currant Road, P.O. Box 869, Fall River, MA 02722-0869.

Subscriptions in the U.S., 12 issues for **$24.00**; in Canada & Mexico surface, **$36.00**; foreign surface for **$44.00**.

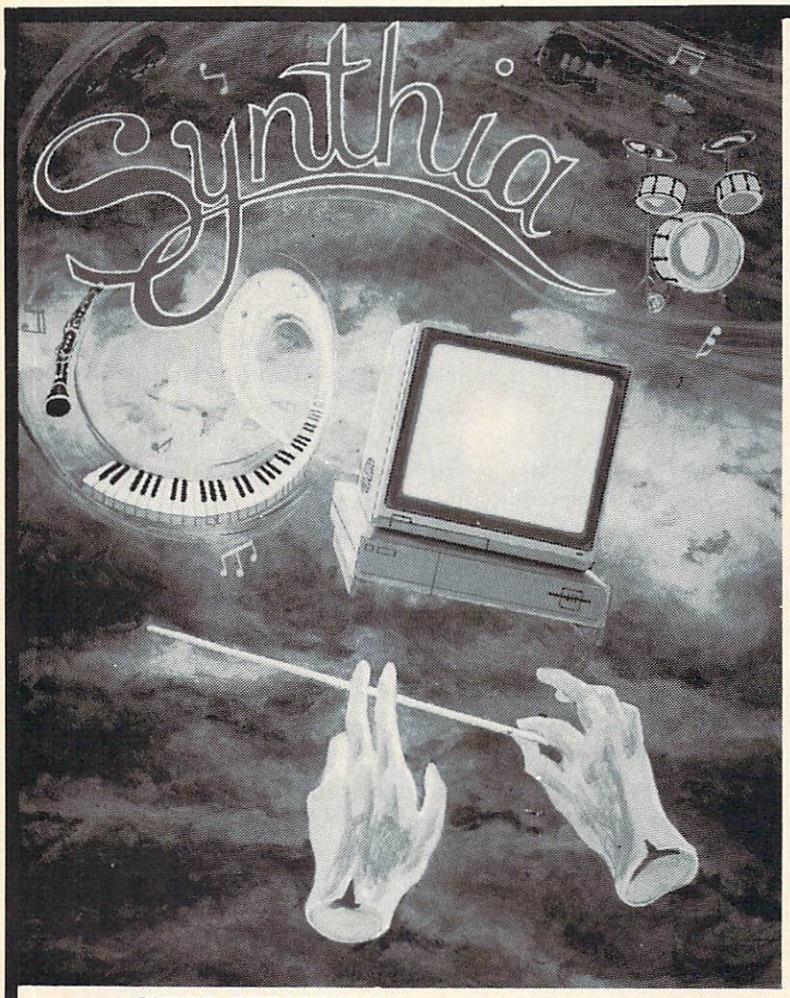
Application to Mail at Second-Class Postage Rates pending at Fall River, MA and additional mailing offices.

POSTMASTER: Send address changes to PiM Publications Inc., P.O. Box 869, Fall River, MA 02722-0869. Printed in the U.S.A.

First Class or Air Mail rates available upon request. PiM Publications, Inc. maintains the right to refuse any advertising.

Pim Publications Inc. is not obligated to return unsolicited materials. All requested returns must be received with a Self Addressed Stamped Mailer.

Send article submissions in both manuscript and disk format to the Co-Editor. Requests for Author's Guides should be directed to the address listed above.



Other Products from The Other Guys

REASON - a professional proofreading system used by universities and writers around the world to analyze and improve writing. **(Has helped raise students grades when used faithfully.)** **$395.00**

OMEGA FILE - a *REAL* data base & mail merge **$79.99**

PROMISE - the *BEST* high speed spell checker. (Even better than Zing!® Spell) **$49.99**

KEEP-Trak GL - general ledger for home or business **$49.99**

AMT - amortization program **$39.99**

MATCH-IT -teaches shapes & colors (preschool) **$39.99**

MATH-A-MAGICIAN - add, subtract, multiply & divide **$39.99**

Call or write for more information.

SYNTHIA *High Performance Digital Synthesizer*

A state of the art music tool which will:

Create digital **IFF Instruments** for use with nearly all music programs!

Modifying existing **IFF Instruments.** Use *SYNTIIIA* on digitized samples to add reverb, wow, and other enhancements.

SOMETIIING FOR EVERYONE:

Additive Synthesis - a traditional method which can create almost any type of instrument.

Plucked String Synthesis - simulates plucked strings . . . right down to the 'pluck'.

Interpolative Synthesis - a method which introduces the natural imperfections found in instruments. (Instruments such as brass, woodwinds, pianos, etc.)

Percussion - build your own drum set . . . create any drum you desire.

Subtractive Synthesis - a simple method of creating instruments.

Special Effects - includes filtering, amplification, phasing, waveshaping, amplitude modulation, real reverb, and . . .

IFF Music Player - powerful and compact. Now you can enjoy those songs that needed a memory expansion before! Up to 32 tracks and 32 IFF Instruments! Supports chords, ties, etc.

NOW SHIPPING

IS IT LIVE . . . OR IS IT SYNTIIIA?

Synthia uses the latest technology to generate realistic sounding instruments and even the new families of instruments sound real. A real synthesizer on a real computer!

Why buy digitized instruments when you can *SYNTIIIAsize* them?

Requires AMIGA 512K

$99.99

THE OTHER GUYS

55 North Main Street
Suite 301-D
PO Box H
Logan Utah 84321

(801) 753-7620
(800) 942-9402

3. Make sure your power supply is firmly connected to the wall and the card.
4. Ground your PALs per the article in AC V3.3. (Don't forget the correction in AC V3.4!)
5. If all this fails, please return the product to Micron Technologies for evaluation and corrective action.

Ms. Shriver was emphatic that there should not be any problem with Micron's card and asked that anyone with such a concern call them immediately.

A New Amiga User Group

Dear AC,
First, I would like to add our Users Group to your list.

Greater Lafayette Amiga Users Group
(GLAUG)
P.O. Box 246
Lafayette, IN 47902

President/Editor	Steve Sinclair
Vice President	Tom Burns
Treasurer/Secretary	Gary Yates
Librarian	Jon Wiggins

We meet at the Married Student Tenant Council Building on the Purdue Campus on Nimitz Drive in West Lafayette. Times are first Saturday of each month for S.I.G.'s, and the third Sunday of each month for regular meetings. We invite all interested persons of the Amiga persuasion to come to a meeting.

Second, I would like to thank you and Stephen Kemp for the "C Notes from the C Group" introductory series on C. I have just started C, and these last two articles have been great. Please keep it up. Two comments: in linking the Sample program in (V3.7 p.91-92), it should have been pointed out that the order of the libraries makes a difference (switch the C and the MA causes the float and double not to be printed); and there was no output shown for the float or double.

Thank you very much for your help and for AC,

Steve Sinclair
Indiana

A Letter of Thanks

Dear AC:
We have been Apple computer users for over eight years and currently have an Apple II plus and an Apple IIgs. We have just purchased an Amiga 2000 for use in our Video Production business and have therefore been avid readers of your magazine for several months. Your articles were very helpful in our decision making process and helped us decide to make the switch from Apple. We are very pleased with our decision.

I am writing in response to your article in the August 1988 issue on the Amiga interface for blind users. My son recently had major eye surgery and, as a result, is now totally blind. He has been legally blind for most of his life, but has managed to get through two years of college with various adaptive aids. Now, however, his needs have changed and he can no longer use his regular computer or word processor. He will be continuing his education as a journalism major at San Diego State University this fall. Needless to say, your article provided a ray of sunshine in what has been a gloomy summer for my son. Please put us in touch with the authors, Mr. Carl Mann and Mr. David Hunt.

Thank you for helping solve a business related problem, but more importantly, for providing hope for a personal problem.

Sincerely,
Jane E. McGinnis
San Jose, CA 95148

Thank you for the kind words. You are the reason we started AC, and you are definitely the reason we continue. It is important to help people, and the Amiga is finding new and better ways of doing this every day. By sponsoring these articles, we reach more people each issue. There is no greater reward for our work then seeing it applied.

ROOMERS Gets Blasted!

Dear Amazing Computing,
A "Roomer" in a recent issue of *Amazing Computing* mentioned that the FrameGrabber's output, as viewed at Spring COMDEX, did not match up to the Quality seen on demonstration disks. This rumor is false. In fact, 99% of all pictures on our demonstration disks were digitized LIVE at the Fall COMDEX in November and recently, the Spring COMDEX in Atlanta. Therefore, the camera and equipment we used to make the demo disks are identical to what viewers actually saw in person at COMDEX.

We hope the facts will discourage any misled readers from believing the so-called "Roomer" that professional video equipment is required to obtain the quality shown on our FrameGrabber demo disks. With FrameGrabber, what you saw is what you'll get.

As for the "Roomers" column, what possible value can AC readers get from false, negative information which is directed towards hurting a company's reputation? What most of us enjoy about "Roomers" is discovering potentially new and exciting Amiga products and/or developments. Any other kind of derogatory comments do not make interesting reading and should be left out of an otherwise outstanding magazine.

Sincerely,
Robert B. Delisa
Colorado

The Bandito agrees with you. On further consideration, it seems rather unlikely that you would not be using your best equipment at a show. Any change in monitors can also make graphics appear differently. He apologized for any inconvenience this error may have caused.

The Bandito's objective to discover the truths behind the falsehoods, not to create them.

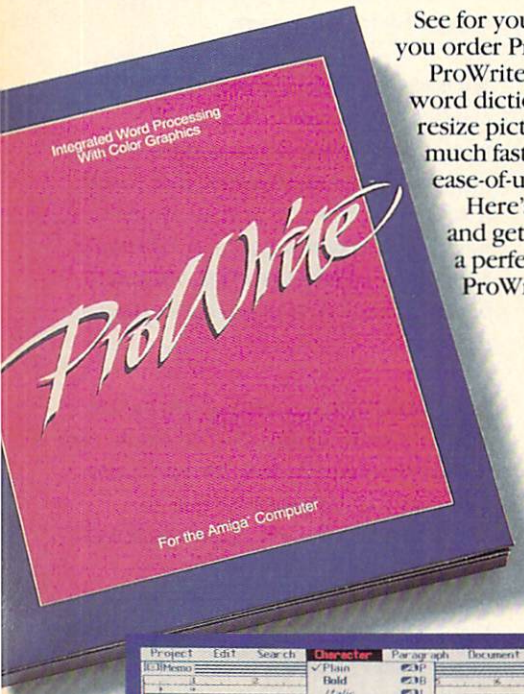
I DEMAND A RETRACTION!

Dear AC:
Yesterday I was on the phone with the customer service operator at Go Amigo, when upon recognizing my name, told me that I had been libled (*sic*) by the Bandito in *Amazing Computing*. Today I received my copy of *Amazing Computing*, and to my horror, found that yes, both Electronic Arts and myself had indeed been libled (sic).

It's Time To See How Your Word Processor Stacks Up To ProWrite™ 2.0

Feature	ProWrite	Scribble	TextCraft	VizaWrite	KindWords	LPD Writer
SPELLING CHECKER	✓	✓			✓	✓
MAILMERGE	✓	✓		✓	✓	✓
OPEN MULTIPLE DOCUMENTS	✓	✓		✓		✓
TRUE MULTIPLE FONTS	✓			✓		
INCLUDE COLOR GRAPHICS	✓				✓	
PLACE GRAPHICS ANYWHERE ON THE PAGE	✓					
USE COLOR FONTS	✓					
WYSIWYG DISPLAY	✓		✓	✓	✓	✓
USER-SETABLE PREFERENCES	✓					
LEFT, RIGHT AND DECIMAL TABS	✓			✓		
PARAGRAPH SORTING	✓					
CHARACTER, WORD, LINE, AND PARAGRAPH COUNTS	✓					
FAST GRAPHICS PRINTING	✓			✓		
USE ANY PREFERENCES PRINTER	✓	✓	✓			✓
AUTOMATICALLY CONFIGURES TO PRINTER	✓					

Now You Can Trade Up To ProWrite And Save $50



See for yourself—trade in your current word processing software, and get $50 off when you order ProWrite, the only multi-font color graphics word processor for the Amiga®!

ProWrite 2.0 has a number of powerful new features. A spelling checker with a 95,000-word dictionary. Mail merge. The ability to read hold-and-modify (HAM) pictures, and to resize pictures as well. In addition, ProWrite has the Workbench 1.3 printer drivers, for much faster and higher quality graphics printing. All this, plus ProWrite's flexibility and ease-of-use combine to make ProWrite the best word processor for the Amiga.

Here's the offer: just send us the master disk of the word processor you're using now, and get ProWrite, version 2.0, for only $75! That's a savings of 40% —which makes this a perfect time to reconsider your word processor. Because now, when you compare ProWrite and the competition, it really pays!

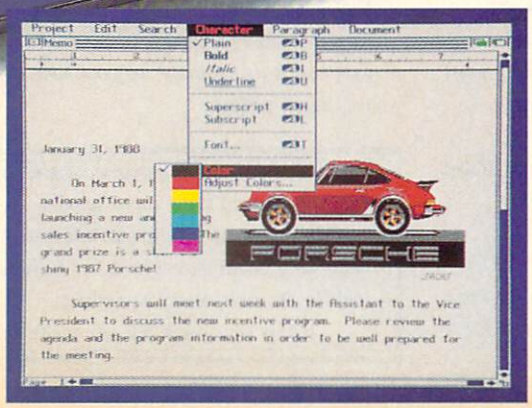
CALL FOR A FREE BROCHURE ON PROWRITE AND FLOW™, THE IDEA PROCESSOR FOR AMIGA.

New Horizons
SOFTWARE

First In Personal Productivity And Creativity.

P.O. Box 43167 / Austin, Texas 78745 / (512) 328-6650

ProWrite is a trademark of New Horizons Software, Inc. Amiga is a registered trademark of Commodore-Amiga, Inc.



AC

I'M READY TO MOVE UP TO PROWRITE 2.0!

Here's my word processor master disk and a check or money order for $75 payable to New Horizons Software, Inc. Send me the new ProWrite 2.0! (Texas residents please add $6 sales tax).

NAME

ADDRESS

CITY STATE ZIP

I demand an apology and a full retraction in the next issue of *Amazing Computing*. You have questioned my integrity along with that of Electronic Arts. Without ever contacting us, you started false roomers. Everyone concerned with the *Computer Chronicles* TV Amiga show knew that Deluxe PhotoLab was secret, and that the only way to demonstrate uses of the Amiga was to work with DigiPaint for the TV presentation. I was showing a current job that I was working on which happened to be the cover of Deluxe PhotoLab. Electronic Arts gave me permission to show the cover in progress, as long as I did not show the beta Deluxe PhotoLab screens.

We had problems during taping because the cover photos brushes did not transfer well to Digi-Paint. That is why I am shown cutting out a brush and moving it around. The Deluxe PhotoLab cover photo could not have been accomplished as shown with Digi-Paint. The photo was especially created to include the use of features only available in Deluxe PhotoLab. Not one pixel was created, changed, etc. using Digi-Paint. That would not be ethical. I am hired not just because of my skills, but my reputation for honesty. You have done me a disservice, as well as the good folks at Electronic Arts.

While you are getting down to "irony of ironies..." think about roomer articles that start rather than just reporting roomers. My attorney Marc Pasin and I are waiting your "Amazing" retraction.

Yours,
Larry Keenan
California

In all honesty, the Bandito was working from the knowledge of your national syndicated television demonstration. The Bandito, as well as millions of others, viewed your performance. It appears highly unfair to both the television audience and the creators of Digipaint that you would be "demonstrating" a machine's capabilities using a program which did not create the art.

The Bandito assumed the work you were demonstrating before the camera to millions of viewers was honestly conceived with the product you were showing. It is a natural, logical assumption.

Your audience was unaware of any secret agreements you had made with Computer Chronicles, *Electronic Arts, or anyone else. They were forced to judge your output by what you were doing on the screen. Perhaps it would have been better for you to have exhibited another graphic you had created using the tools you demonstrated on the show. This would appear a more rational view.*

The Bandito offers an apology for making an incorrect assumption based on the facts.

As for AC, we have in the past, and currently do, run a disclaimer that the material enclosed is for entertainment only. Unfortunately, the issue you were mentioned in did not contain this disclaimer. (A fact that has been made ***PAINFULLY*** *clear to our editorial staff, which has promised faithfully* ***NEVER*** *to run "Roomers" again without this disclaimer.) We apologize for any inconvenience this article may have caused.*

OXXI, Inc. is not MaxiSoft.

John Houston of OXXI, Inc. telephoned at press time to inform us that the Bandito was incorrect in his column of AC V3.8. Mr. Houston stated the following: "Outside of publishing MaxiPlan, OXXI, Inc. has no association with MaxiSoft, EA, or their suit."

Well friends, this makes three complaints against "Roomers" this month. This does not include the phone calls I received which were not followed by the complaint letters I requested (and I always ask everyone to write us). Nor does this include the attacks I have received from Amiga personalities at trade shows.

The complaints range from, "That didn't happen!" to "We did not want that released."

To begin, let me say that I am not here to defend "Roomers." As an editor, I find the idea of a column written by an anonymous third party which can not be verified not to be true journalism. This is why the column is called "Roomers," and this is why our disclaimer should (it had better!) appear in each issue.

Some of the column's detractors readily point out that each item could be verified by the AC editorial department. In response, let me make two points. First, if we verified the rumors, they would no longer be rumors, but rather facts that should be published in "Hot On The Shelves" or another article. Second, when some such verifications are attempted, we either receive no information, or the information is denied followed by an explanation which restructures the original information to be favorable to the company.

This places us at an extreme disadvantage. At that point, we would be allowing the vendors and advertisers to write our copy. While this procedure may be acceptable to some, it would not be acceptable to our readers.

We have been asked to keep the rumors to product announcements only. This request was followed by a comment by an individual who cornered me at a recent Amiga exhibition. He said we should never pre-announce products, because it severely damages sales of the company's product and other companies' current products.

AC is faced with a variety of choices and none are easily acceptable. AC would like to hear from the readers. We want your input on how "Roomers" should be handled, and what you expect from this type of article.

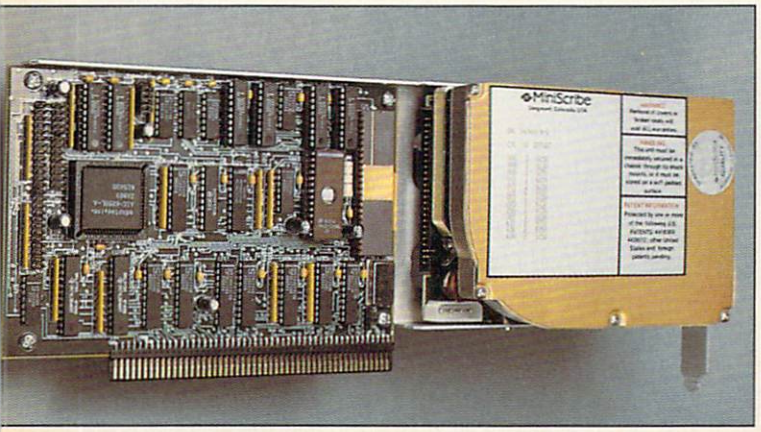
From the beginning, AC has been the Amiga user's forum. It is here we discuss the things that matter to you. Get involved, write!

We welcome your comments

All AC readers who have letters, questions or comments printed in AC receive a certificate good for 5 free Public Domain Software disks.

Keep involved. Please write us!

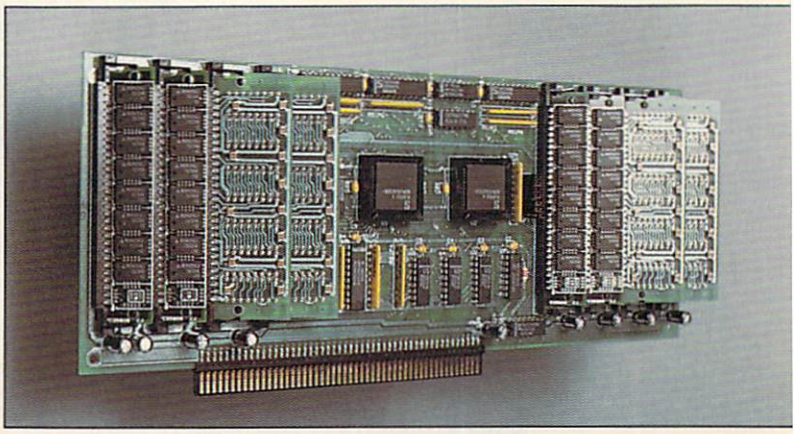
HardFrame/2000
The Super-speed, DMA, SCSI Hard Disk Interface with 1.3 Autobooting



How fast is fast? **HardFrame/2000** transfers data at Amiga bus speeds! It's actually faster than the hard disk mechanism itself! And even more important in the Amiga's multitasking environment, **HardFrame/2000** has extremely efficient DMA circuitry to get on and off the bus in almost no time at all: 280ns to get on; 200ns to get off. **HardFrame/2000** autoboots under AmigaDOS™ 1.3 and is fully compatible with the new Fast File System. The core of any DMA SCSI interface is its SCSI protocol chip and DMA chip. MicroBotics has chosen the new, high performance Adaptec AIC-6250 SCSI chip, capable of up to 5 megabytes per second raw transfer speed, and the Signetics 68430 DMA chip running at 12.5 megahertz. Then we added additional FIFO buffering and enabled 16-bit wide data transfers for maximum throughput. The sophisticated design of **HardFrame/2000** provides for automatic SCSI arbitration, selection and reselection. The hardware supports either synchronous or asynchronous data transfer. **HardFrame/2000** can function as either the SCSI bus initiator or the target and can reside in a multiple master environment. Physically, **HardFrame/2000** is optimally flexible: the compact, half-size card comes attached to a full length, plated aluminum frame. The frame has mounting holes positioned to accept standard, 3.5" SCSI hard disk units such as those manufactured by MiniScribe, Seagate, Rodime, and others (hard disk mechanisms must be supplied by the user or his dealer as a separate purchase item). Alternatively, you can cable-connect to a SCSI drive mounted in your Amiga's disk bay or in an external chassis. As many as seven hard disks may be connected to a single HardFrame. There is no size limit on each disk. **HardFrame/2000** includes a 50-pin SCSI cable and header connectors for either 50-pin or 25-pin cable connection. Also included is a current tap to power frame-mounted drives directly from the slot itself. **HardFrame/2000** comes complete with driver, installation, and diagnostic software. Available September 1988. Suggested list price, $329 (hard disk not included).

The HardFrame/2000 photo shows the product with a MiniScribe 20 megabyte hard disk installed. Hard disks are not included in the purchase price of HardFrame. Note that if placed in the first slot, HardFrame uses only one slot.

8-UP! The Eight Megabyte Memory Card with Amiga™-specific DRAM Controller Logic



All the memory space you and your Amiga 2000 need –in a modern, highly integrated FastRAM expansion board. In **8-UP!**, MicroBotics went all the way to provide you with a truly Amiga-specific memory design to meet the special demands of the Amiga's high speed multitasking environment: The heart of any memory expansion is its *DRAM controller circuitry*. Rather than compromising with off-the-shelf parts, MicroBotics developed its own, custom controller design and built it into high-speed, *Programmable Macro Logic* chips (Signetics PLHS501). These new, super chips (each **8-UP!** uses two PML's) permit MicroBotics to employ *sparse refresh* technology to assure that your **8-UP!** is a truly zero wait-state/minimal-refresh-collision memory design. If you're putting eight megabytes in only one slot, that means that you probably have plans for your other A2000 slots. **8-UP!** gives you new freedom to do that planning since, unlike other ram peripherals, it is an extremely low-power memory card– a single, fully-loaded, 8-megabyte **8-UP!** draws an astoundingly efficient 0800 milliamps! That's less than *two-fifths* of the power "budget" for a single slot! Low power draw also means that the card is cool-running for reliability and long life (not to mention a cooler Amiga!). **8-UP!** offers you maximum flexibility in memory configuration: it is organized into two separate *PIC's* (Amiga-speak for autoconfiguring peripherals). Each **8-UP!** PIC consists of four SIMM module sockets; these sockets accept either 256k-byte or 1 megabyte *SIMM's* (Single Inline Memory Modules). You can also purchase optional *PopSIMM* boards from MicroBotics; fill them with conventional RAM ; then use PopSIMM's to fill your **8-UP!** The card can run with as little as 512k of memory or as much as eight megs –with many intermediate configurations possible (particularly the six megabyte configuration, most desirable for use with a BridgeCard™). **8-UP!** is speedy, efficient, custom memory technology for your Amiga 2000 –and it's available now! **8-UP!** suggested list price is $199 (0k installed). Optional **PopSIMM's** are $49.95 per pair.

The 8-UP! photo shows the card half populated with conventional SIMM modules and half with MicroBotics PopSIMM's. PopSIMM's (without DRAM installed) are available as separate purchase items.

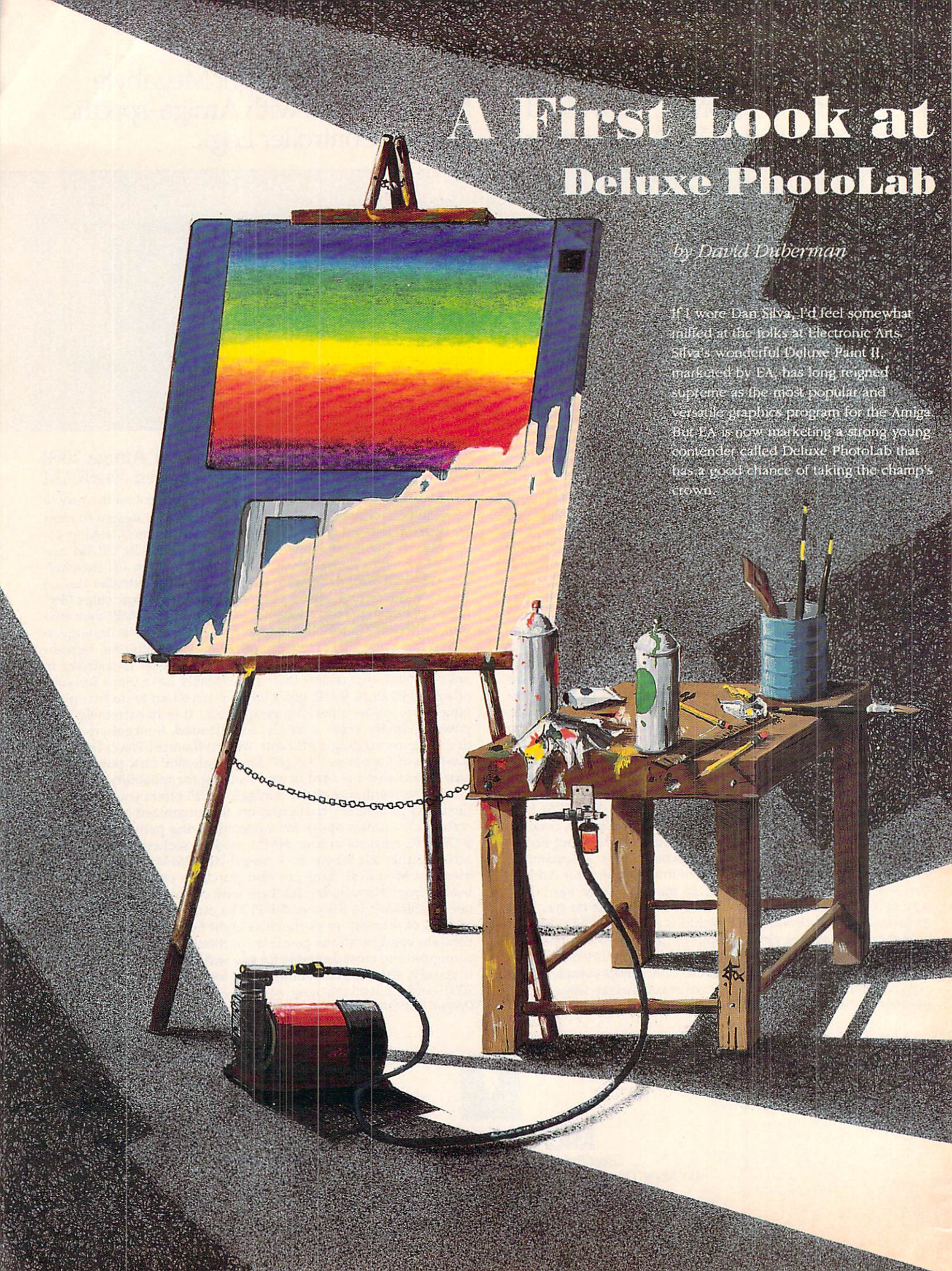
MicroBotics, Inc.

Great Products Since the Amiga Was Born!
811 Alpha Drive, Suite 335, Richardson, Texas 75081 (214) 437-5330
SOLD ONLY THROUGH YOUR AMIGA DEALER
Tell your dealer he can quick-order from MicroBotics directly – no minimum quantity –show him this ad!

A First Look at Deluxe PhotoLab

by David Duberman

If I were Dan Silva, I'd feel somewhat miffed at the folks at Electronic Arts. Silva's wonderful Deluxe Paint II, marketed by EA, has long reigned supreme as the most popular and versatile graphics program for the Amiga. But EA is now marketing a strong young contender called Deluxe PhotoLab that has a good chance of taking the champ's crown.



Deluxe PhotoLab is comprised of three programs: Paint, with which the bulk of this review is concerned; Posters, a powerful graphics printing program; and Colors, which gives you some useful and interesting image-processing capabilities not present in Paint.

Paint by no means replicates all DPaint's functions—missing are the latter's perspective mapping, stencil, color cycling, grid and mirror draw, and special brush mode effects. But Paint's ability to work in any standard Amiga graphics mode (including hold-and-modify (HAM) and Extra-Half-Bright) make it a more versatile program. (Extra-Half-Bright mode is available only on recently manufactured Amigas.) Of course, HAM mode lets you use any of the Amiga's 4096 colors anywhere on the screen, with slight restrictions that are well-explained in the excellent user manual.

Paint is unique in that it lets you work on different pictures in different resolutions at the same time! If you copy blocks between pictures, the program automatically converts the picture to the different graphics mode.

Which brings me to Paint's other major improvement over DPaint: the ability to use all available memory for pictures. Most Amiga graphics programs limit the amount of loaded graphics data to the 512K maximum imposed by chip RAM. Paint lets you simultaneously load as many different graphics screens as your computer's total memory allows—not just the first 512K. You can also create pictures as large as memory allows, well beyond the screen borders. You can switch between pictures with a press of a key. Paint doesn't let you swap pictures between screens as easily as DPaint—you must Cut and Paste.

Whenever you start Paint or create a new screen within the program, you're presented with a requester to set the graphics mode. Your choices for horizontal resolution—which determines the number of colors available for painting—are Low Resolution, High Resolution, Extra Half Bright, and Hold and Modify. Set vertical resolution to Interlace or Non-Interlace.

The Depth gadget further controls the number of colors available by letting you set the number of bit planes used in the picture. The Size gadget offers the option of using the full screen or a 3/4- height screen for memory conservation. Overscan isn't an option in the initial setup, since you can set the screen to any size while painting.

The drawing screen appears next with a menu bar across the top and a horizontal toolbox bar immediately under it. Despite the 90-degree reorientation, the toolbox closely resembles DPaint's. (I don't think EA should sue itself for look-and-feel!)

Most familiar tools of the digital artist are here: dotted and continuous freehand draw, straight line, text, rectangle, polygon, and oval (circle), airbrush, fill, cut (custom brush), magnify/zoom, and the fixup team Clear and Undo. The one major deviance from the DPaint standard involves the curve tool—in DPaint it creates parabolic arcs; in Paint it draws an S-curve (or any other curve defined by four points).

The toolbox bar also contains the working color palette. The palette sets drawing and background colors and operates identically to palettes in most Amiga paint programs. (The number of available colors is determined by graphics mode and the number of bit planes.) You can set colors and create spreads across ranges by flipping to an alternate half-height Palette screen.

The Palette screen also gives you the Paint Set, a group of 128 additional slots for Palette colors that you can load and save separately. The Paint Set is particularly useful for creating smooth ranges of color for use in gradient fills. In HAM mode, you can use these spreads immediately, but in the other modes, you must copy the colors into the standard palette. Paint's Palette requester is a powerful tool for manipulating color.

Paint's power resides in its menus. The Project menu contains the standard Load and Save commands. Also included are Load At, which lets you specify alternate loading coordinates for combining pictures, and Save At, which lets you save any part of a picture. Paint uses an unusual file requester in which files, directories and available volumes are all listed in a single vertically-scrolling window. To change to a different disk or drive, you must scroll to the bottom of the window (or press the V key) and select the volume's name. This arrangement easily accommodates any number of volumes, but in a multi-volume situation, I much prefer the drive button gadgets (DF0:, DH0:, etc.) .

Page Size, another important Project menu item, lets you expand the area available for painting horizontally and vertically to within total memory limits. You can use the arrow keys to scroll to different parts of the picture or jump around with the Show Page command. In HAM non-interlace mode, on a one-megabyte Amiga with no other programs running (except the Workbench), Paint allows a single picture to be as large as 64,000 square pixels (800 x 800 pixels). The Project menu's powerful Print command lets you size the hard-copy image proportionally or non-proportionally in pixels, inches, or percentage of the page size. Other options are normal or sideways orientation, shading, and centering.

The Brush menu performs standard manipulations on picture sections picked up with the Cut command, including Load, Save, Rotation, Print, and Resize. Resize is most effective because you can use two sets of crosshairs to specify the brush's new size anywhere on the screen—ideally, of course, where the object is to be placed. Resize Draw lets you combine resizing and placement in a single operation.

(continued)

The Handle command lets you specify an offset between the brush and mouse pointer location, and Remap performs an accurate translation of palettes between a brush and a picture from different origins. I have a lot of fun with the Grab Last command which picks up the most recently drawn object as a new brush. If you hold down the Shift key while "grabbing last," you get what was underneath what you drew. This is useful for creating multi-color text.

The Modes menu contains only three items, but holds much of Paint's image-manipulation power in the Paint Modes submenu. But first to the Brush Mode submenu, which offers Matte (standard brush), Color (one-color painting using the current brush shape), Pattern (lets you draw in the current brush pattern with any tool such as line or airbrush), and Store (gives an irregular-shaped brush a rectangular outline).

The Paint Modes submenu offers a wealth of picture-combining options, most of which work best in HAM mode. Proper palette selection lets you use commands like Blend for effective semi-transparent effects in any graphics mode. Available modes include Solid, Low Mix (applies the lightest amount of brush color), Mix (applies color at one-fourth strength), Average (applies color at half strength), and Blend (adds color at three-fourths strength).

Other modes are Shade (discussed below), Subtract Picture or Brush (subtracts color values), Scale (gives embossed effect), Scale2 (affects picture contrast), Add (sums color values), and statistical operators Max and Min. Also available are the logical operators XOR, OR, and AND.

The HLF and B&W modes remove half or all the color from the affected area. The manual offers examples of calculated RGB numerical values for all effects, but correctly suggests that you get a better idea of how they work by trying them out. The manual also offers several useful tutorials that show practical uses for some of these effects.

Finally, the Modes menu's Affect setting determines whether subsequent paint operations affect the entire drawing (the normal setting) or only the foreground or background.

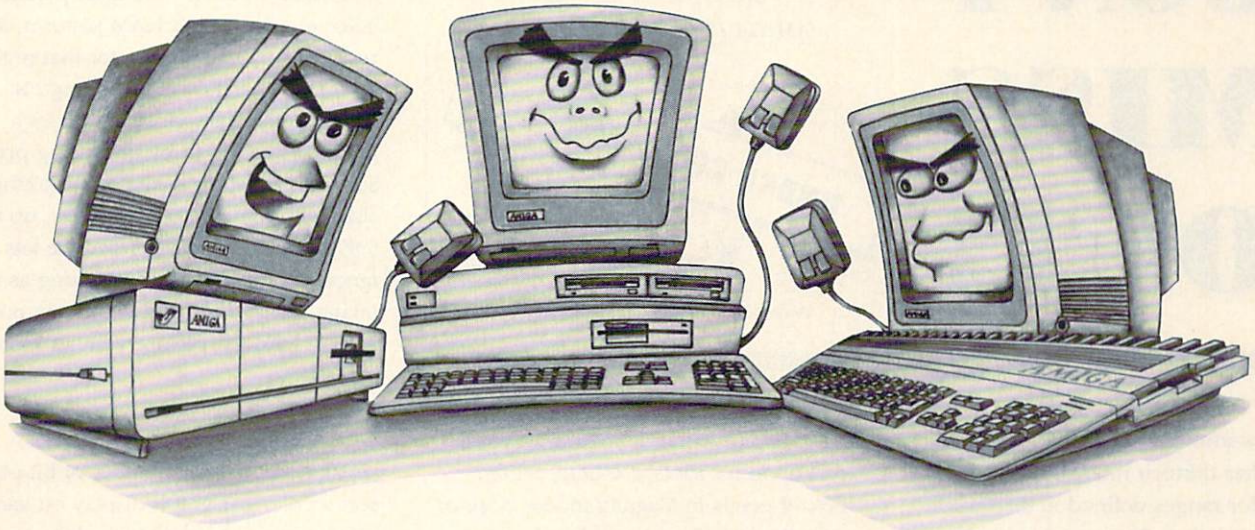
The Options menu specifies a number of settings used with other commands and menu settings. The Repeat command repeats the last drawing action with the current settings (handy for experimention when used with the Undo command).

The Shade Control requester in the Options menu gives you a number of settings for use with the Shade Paint mode. Shaded brushes typically vary in strength across their breadth and width, as if illuminated by a point light source above the center. However, Paint lets you offset this highlight to any point on the brush, or to use a vertical or horizontal highlight instead. You can even reverse the fall-off effect for brushes highlighted at the edges, instead of at the center. You can also set the dithering between bands of color in solid-color shaded brush.

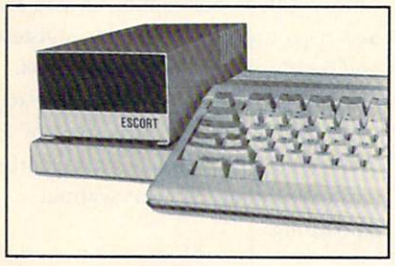
The Fill Control's four basic settings are Solid Color, Brush Pattern, Trace Edges, and Gradient. These affect Fill and Filled Shape operations. The Fill Offset gadget lets you specify any offset for Brush Pattern fills. For example, you can line up the edges of the brush and the filled shape. The HAM Closeness gadget, a new and useful feature in HAM paint programs, lets fill operations affect pixels that are close in value to the area initially selected for filling. In a single operation, you can vary this setting to fill similarly-colored areas .

(continued on page 14)

POWER HUNGRY

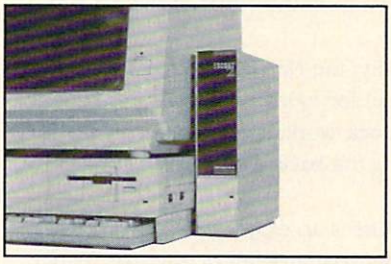


Whether you own an Amiga 500, 1000 or 2000 you'll want to feed it right. Fortunately we're here to help. At Expansion Technologies we provide high quality expansion products that are guaranteed to keep your Amiga growing happy and strong. Just check out our menu :



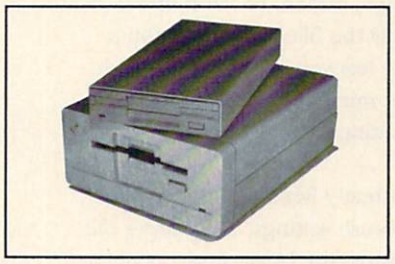
If your Amiga has a really voracious appetite, but your pocket book is on a diet, try our Escort 30Mb SCSI hard drive at a 20Mb price. Controllers, with a bus return are available for both the A500 as well as the A1000. Each accepts our 3.5 or 5.25 inch drive chassis (available in sizes up to 80Mb) with fan and power supply and is fully compatible with WorkBench 1.3. If you prefer buffet, buy the kit and supply your own drive. However you do it you'll enjoy lots of storage space at a great price.
Now thats serious fast food!

Escort Hard Drive kits
A500 30Mb $699
A500 Kit 3.5 inches $339
A1000 30Mb $759
A1000 Kit 3.5 inches $379
Call for other sizes.



For the discriminating pallet of the expansion hungry Amiga 1000 we have the ever popular Escort 2. It's 2 megabytes of no wait-state memory that meets all the known standards and then some. It's also a uniquely designed vertical 2-slot card cage that offers incredible flexibility. Like the ability to upgrade to a scrumptious 4 megabytes of memory, or you can add a hard disk controller that uses IBM type hard drives, or an external power supply for the bus return or...well, you get the idea.
Truly a meal fit for a king!

Escort 2 (0k mem).....$339
2 megabyte upgrade card (0K mem).....$279
ST-506 hard drive controller card and power supply $399



Simply the best engineered 3.5 inch external floppy drive available for your Amiga, and it's 100% 1010 compatible. The precision drive mechanism is ultra quiet and has the highest MTBF specifications. A low power CMOS design and pass through allows for easy daisy chaining too. Its low profile (1x4x8 in.) allows A500 disks to eject over the top, and it fits right under an A1000. Includes an extra long cable, heavy steel chassis and faceplate with dust cover. Why buy less than this Made In The U.S.A drive.
Now you're cooking with gas!

Escort 3.5 in. floppy $179

Feed your power hungry Amiga right- give it peripherals from Expansion Technologies.

** Toothpicks not included*

VISA and MasterCard accepted. Dealer Inquiries Invited.

EXPANSION
TECHNOLOGIES

46127 Landing Pkwy, Fremont, CA 94538 415-656-2890

The powerful (and amazing!) Gradient setting has thirteen modes, seven for use with color ranges defined in the Palette requester and six for use with custom brushes. Range fills can be uniform vertical or horizontal. Each color band's size is determined by the height or width of the page, screen, or filled object, or it can follow the filled area's contours. Dithering lets you soften color bands' edges, or mix them up thoroughly at higher settings.

Gradient really flexes Paint's muscles with its brush settings. The border fills option warps the brush horizontally to match the filled area's contours, but leaves the vertical format untouched. Horizontal, Vertical and Stretch magnify the brush uniformly in the indicated direction(s) to match the filled area's size.

Brush Pattern uses dithering to randomize each copy of the brush, for such effects as irregular or fuzzy but recognizable patterned backgrounds. And most spectacularly, Brush Warp reshapes the brush to fit the filled shapes contours, resulting in some interesting distortions. For example, you can create a fish-eye effect by warping a rectangular brush onto a circular shape.

Other Paint menu commands let you load up to six fonts at a time, set the look of pixels in Magnify mode, close or open the Workbench, set brush background to transparent or opaque, set smoothing (anti-aliasing) for brush resizing, and indicate whether ovals are drawn from the edges or the center.

Finally, the No Background command is useful for touching up pictures in genlock work, since it prevents you from using the background color in painting.

Posters is an easy to use program that prints your artwork in any size. You're presented with a grid of vertical 8 1/2" x 11" pages—sixteen across by eleven down. When a picture to print is loaded, an outline showing relative size and position appears in the upper left corner of the grid. You can interactively resize the image by dragging the outline's lower right corner, or by clicking on arrow keys. You can also reorient the image horizontally and resize the standard page size for different sized printers. (The program takes its standard settings from your Workbench preferences.) If you wish, the program preserves the picture's original aspect ratio when resizing. If your printout size is larger than the page size, the picture is printed across multiple pages as indicated by the grid. Cutting and pasting is necessary to assemble the printed poster.

The Colors program provides a number of powerful image-processing functions that affect an entire picture. The Reduce command frees up little-used palette color registers. With HAM pictures, it replaces any pixels used for that register with HAM pixels and the change is barely noticeable. Mosaic reduces a picture's resolution by increasing pixel size. Resize lets you increase a picture's size, within memory constraints, up to 9,999 pixels square. Resize Save lets you ignore memory limits by resizing as the image is saved to disk. (You may not be able to load the picture without more memory).

Other commands let you do color separations, set the number of bit planes, sort a color bar graph display on various criteria, remap a picture's palette to another picture on disk, remove all color, or convert the picture to a negative.

Colors' palette commands let you swap and copy colors, swap color registers, and meld registers in various ways. Finally, you can convert a picture to any other graphics mode supported by Paint, with or without a palette restructuring and resizing and with or without smoothing.

It's hard to go wrong with Deluxe PhotoLab as an all-around paint program/image processing tool unless you require the special capabilities provided by other paint programs (such as DPaint's perspective mapping or Photon Paint's 3D texture mapping). Deluxe PhotoLab is truly the first all-around graphic artist's tool for the Amiga, and it pushes the machine to its limits in many ways. (Shown by the long waits for many operations.) The program seems well tested—it hardly crashed at all—and all functions work as advertised. Other paint programs may cost less, but none provides the breadth of graphics power present in Deluxe PhotoLab.

•AC•

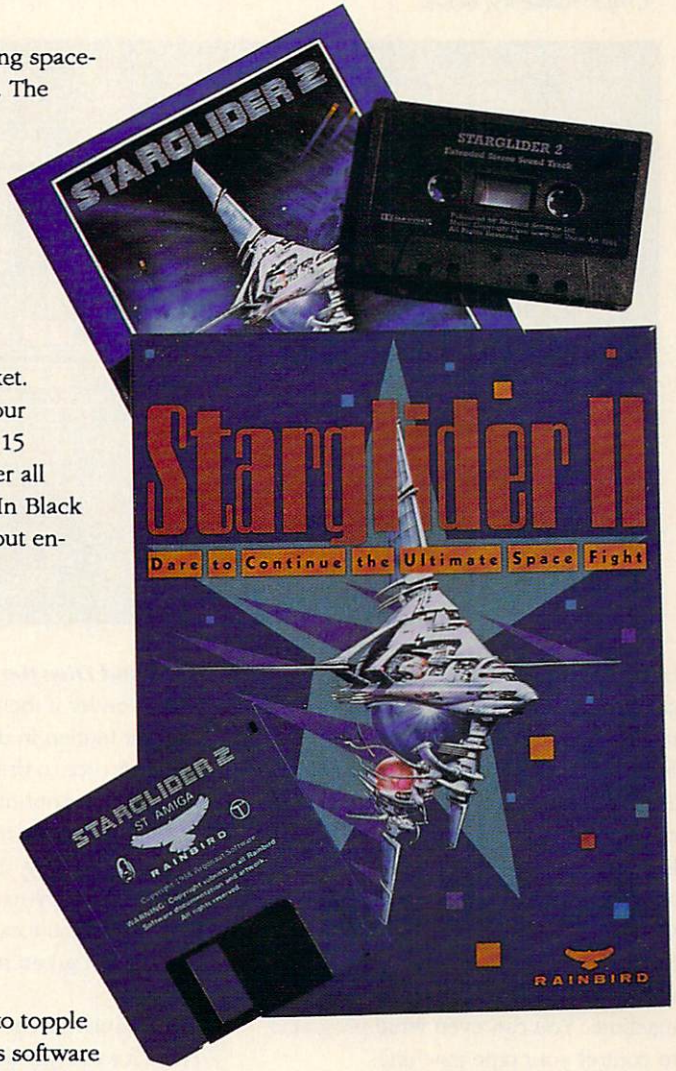
Hot on the Shelves

by Michael T. Cabral

Glide Again!

Remember the empirialistic Ergons you triumphantly blasted into hyperspace in the original Starglider? They're baaaack. **Starglider II** plops you in the cockpit once again against the energetically evil Ergons. This time you face a revamped fleet bent on pocketing your homeland and making a crispy Novenian out of you. With your "state-of-the-22nd-century" winged weapon, you've got at at least a chance as you hop from planet to planet desperately trying to defend blessed Novenia.

A three-dimensional control panel lets you keep one eye on oncoming spacecraft and uglies, while the other eye aims on dealing out destruction. The realistic flight offered by the lightning-paced solid 3D graphics is enhanced by digitized speech and an earful of sound effects. And as if the on-screen action isn't enough, the game also comes packed with a 44-page novella and a stereo soundtrack on audio cassette. If you've got the guts to continue "the ultimate space flight," Rainbird's got the game.



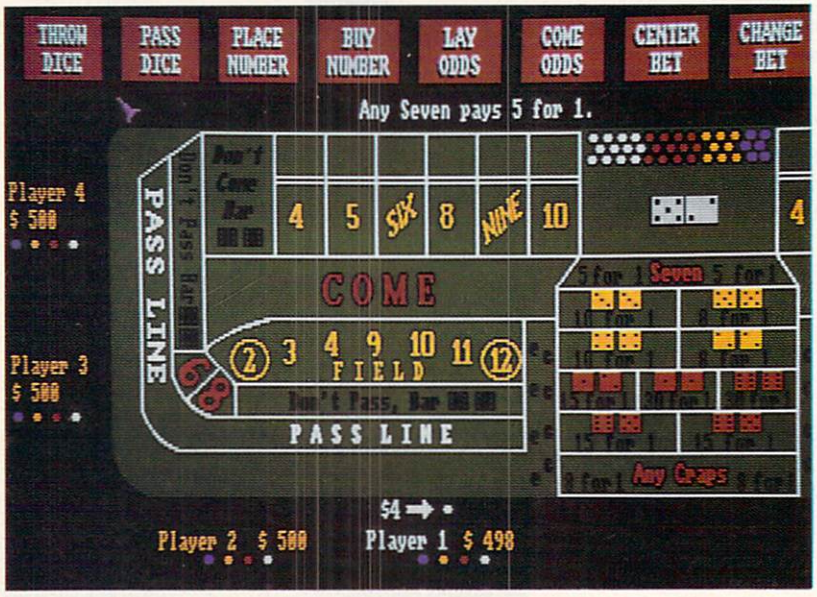
When it "Rains," it pours! Along with Starglider II, Rainbird has also sent **Carrier Command**, **Enlightenment**, and **Black Lamp** to market. Carrier Command puts you at the helm of four aircraft carriers and four amphibious vehicles for a crash course in 3D strategic warfare. With 15 skill levels and 32 different spells, Enlightenment forces you to muster all your conjuring magic to toss an evil wizard out on his bag of tricks. In Black Lamp, a hot-tongued dragon breathes down your neck as you seek out enchanted lamps in a fairy tale-style adventure.

Starglider II $44.95
Carrier Command $44.95
Enlightenment $24.95
Black Lamp $24.95
Rainbird Software
3885 Bohannon Dr.
Menlo Park, CA 94025
(415) 322-0900
(800) 227-6900 (orders)

Photon Control

With the **Photon Video Transport Controller**, Microillusions tries to topple the wall between Amiga graphics and animation and video tape. This software gives you full control over your video tape controller , allowing both frame by frame and continuous segment recording of animation.

Photon Video Transport Controller above, and, Craps Academy, below.



For smooth animation sequencing, you can program the Transport Controller to record anywhere from 1 to 300,000 frames each time an image is displayed. The TimeLapse module allows you to create time lapse sequences. An editor lets you make short, real-time Amiga animations into longer sequences. IN and OUT edit points are settable by mouse or keyboard, and you have unrestrained manual control of your video tape machine. You can even write programs to control your tape machine!

Most low-cost video controllers are supported by Transport Controller, including Lyon Lamb's MINIVAS and VAS IV, BCD's model 2500, Pico System's Animation Controller, and Video Media's V-LAN system.

Photon Video Transport Ctrlr $299.95
Microillusions
17408 Chatsworth St.
Granada Hills, CA 91344
(800) 522-2041

Slice and Dice the Odds

Ever wonder if there's a hot Vegas shooter hidden in the dark side of your soul? It's nice to dream about needing a Brinks escort out of a conquered casino, but who's got the money to find out? **Craps Academy**, the latest in Microillusions' cunningly-named Micro-Vice Series, lets you toss the dice without tossing real green stuff.

This simulation for 1 to 4 players covers craps for everybody from the bankrolled novice up to the Vegas vet. If you don't know a "come out roll" from a "horn bet," the Academy can help. A complete glossary of craps "buzz words" is included, and pop-up help screens fill you in on table layout, handling your chips, making your bets, and even collecting your winnings!

Once you've got the basics down, Craps Academy moves you along to bottom line, cash-in-pocket factors like house percentages, betting systems, money management, and optimal bet sizes. Sucker's bets and bettor's bets are sorted according to house advantage. Table rules can be configured to craps variations found in Vegas, Atlantic City, or Reno. And if you've already adopted a favorite casino, you can set individual rules to fit the rules of your choice table!

Craps Academy $39.95
Microillusions
17408 Chatsworth St.
Granada Hills, CA 91344
(800) 522-2041

Map It Out

Pick up a color-splashed *U.S.A. Today* almost any day and you see examples of choropleth mapping. A typical choropleth might show states with the highest percentage of Democrats in light blue, states with a moderate percentage of Dems in a deeper blue, and states with few Dukakis rooters in black. The choropleth method simply takes any quantifiable variable and plots out distribution by color. The result is an easy-to-read map with a main point that can't be missed.

Choromap, by Bassett Geographic, lets you make choropleth maps on your Amiga. The choropleth process combines three elements: a base map, a list of data for each area unit, and specifications about how to put together the map. Choromap uses three programs—Digitizer, Datamaker, and Mapmaker—to layout choropleth maps.

Digitizer allows you to create an outline of your map using only your Amiga and a transparent acetate sheet. Just trace out your map on the sheet, place it over

your monitor, and mouse-click on all vertices affected by connections and changes in direction. Digitizer then stores the map boundaries as a series of x and y coordinates. Datamaker, your data storehouse, accepts all data as positive integers or as decimal fractions altered by standard arithmetic operations and a constant. After some minor clean up and categorization, Mapmaker takes the data and map information and spits out an appropriately colored choropleth map.

Now it's easy to create an accessible reference of who or what is distributed where in your region.

Choromap
Price unavailable
Bassett Geographic
1103 Rudd Ave.
Auburn, AL 36830

Are You There? Boom!
Not too long ago, Scott Lamb and Software Terminal ushered some fun into the telecomputing world with TeleGames, a batch of classic strategy games contestable via modem. Sure it's great to be able to play chess, checkers, and backgammon by modem, but how about more excitement? You know—a little blood, some guts—all the usual fun. The folks at Software Terminal have come up with an explosive answer to this plea for a bit more spice with **TeleWar**, the blast 'em up follow-up to TeleGames.

TeleWar is a one-half strategic, one-half "blast 'till you drop" war simulation you can play over the same phone lines you talk to Grandma on! Compatible with any modem set at a baud rate of 300 to 9600, TeleWar lets you battle an opponent thousands of miles away. With a null modem connect, you can also hook up two Amigas for simultaneous play against a local enemy squadron leader.

As you move through the twelve war scenarios, rat-a-tat-tatting along with your animated artillery, your opponent sees all your moves graphically enacted on his screen. 3D perspective terrain maps let you keep close watch on advancing enemy forces, and digitized sound effects add all the noise needed for your average battle scene. And if you get tired of blasting bodies on the same terrain all the time, three different maps are available including such unappetizing sites as Marshland Bog and Desert Dune.

TeleWar $39.95
Software Terminal
3014 Alta Mere Dr. Highway 183
Ft. Worth, TX 76116
(817) 244-4150

Supra 2000
The hard drive heavies at Supra Corporation have set their latest creation loose. This time they're out to make your Amiga 2000 into a memory rich speed demon with the SupraDrive for the A2000. The new bambino in the SupraDrive family is a high-performance hard disk with autobooting and quick access time available in 20, 30, and 60Mb.

SupraDrive 2000 features true DMA access, an external SCSI port, and compatibility with RAM boards, digitizers, the Bridgeboard, and all MS-DOS partitions. The drive is available in a plug-in external form or as an internally mountable unit. The internal set-up comes with a DMA interface, the drive and controller, screws, and safe, easy-to-follow installation instructions.

If you want to mount an internal drive other than the Supra and need those same simple directions, Supra Corporation also offers the Supra Interface Kit as a separate package. Included are all Supra's formatting and utilities software and CLImate, a favorite utility of users striving to cut CLI use. Controllers in the stand-alone kit are optional.

SupraDrive 2000
20Mb $699; 30Mb $799; 60Mb $995
Interface Kit $249.95; $399.95 w/ ctrlr
Supra Corporation
1133 Commercial Way
Albany, OR 97321
(503) 967-9075

Friendly Fine Print
For a change, here's something nice in the fine print. In fact, **Fine Print** is a new print utility by Designlab that focuses on sharpening the shades of gray in your dot-matrix printouts. The program beats your average Amiga printouts to the resolution punch by actually building up faint layers of ink on the page.

Rather than fooling your eyes with simulated gray shades and coarse patterns, Fine Print lays down detailed gray shades in a very fine pattern to achieve near-photographic prints. A sample printout of the same IFF file, shot out of the same printer with the same density printer drivers, showed drastic improvement when printed with Fine Print instead of Deluxe Paint.

Strangely but sensibly enough, Fine Print needs a worn out ribbon to reach the right level of detail. The wear allows the layers to be dropped one over the other without saturating the paper. A fresh, dark ribbon creates total blackness in only a few hits, while a ragged ribbon can set down as many as fifteen separate layers for knockout clarity and detail. You can adjust the ink level individually for each color in your original image.

Fine Print is not size finicky either, as it prints both large and small images and can scale any IFF image to any size print. IFF images with up to 32 colors work fine with the program, but only black and white output is possible since only dot matrix printers are supported. A series of Fine Print-specific printer drivers support most popular dot matrix varieties, including Apple's ImageWriter, IBM's Proprinter, Okidata's ML series, Star's Gemini group, most Epsons and a heap of others.

Fine Print
$49.95
Designlab
P. O. Box 419
Oswego, NY 13827
(607) 687-5740

•AC•

Multiscan Monitor Comparison

Magnavox Multi-Mode 8CM873, Mitsubishi Diamond Scan AUM-1371A, Thomson UltraScan 4375M, Logitech AutoSync, and Amiga A-1080

by Steven L. Bender

When the Amiga 1000 came to market in late 1985, very few reasonably priced (under $2000) monitors readily accepted the machine's analog RGB signals. The companion Amiga A-1080 monitor is capable of reasonably good performance, but has its defects in the areas of dot pitch, video bandwidth, and unsaturated colors. A number of other monitors now accept our analog RGB video signals and multiple scan frequencies.

How Color Monitors Work

Three electron guns emit invisible beams of electrons toward the "flat" surface of the CRT screen. These beams are responsible for the red, green, and blue that make up all color combinations. The electron beams pass through a fine shadow mask that has many fine holes which either block the electron beams or lets them pass. The impact of the beams which don't pass through heats the shadow mask, causing changes in the mask's overall shape after the monitor is turned on.

The three electron guns are precisely aligned and angled slightly differently, so the electron beams pass through different holes in the shadow mask and strike different areas on the screen. The beams that pass through continue and strike the phosphor layer on the CRT screen. The phosphor emits visible light which we see as graphics. When the shadow mask is faulty, the beam is misaligned or misconverged and color fringing occurs.

A properly manufactured and aligned shadow mask is critical to a monitor's operation. The shadow mask changes shape slightly as it heats up from the impact of the blocked electron beams. This causes variance in the convergence during the first 20-30 minutes after turn on. The unit which shows misconvergence early on might clear up after 15-20 minutes. Conversely, a well-converged screen might become worse after a while. The monitor's tested here were judged after a sufficient arm-up period.

Multiscan (multiple frequency scan) monitors are reasonably obsolescence proof. These monitors contain much expensive circuitry. This design increases the overall cost, but enables these monitors to work with many machines and many different video standards. It also means that, at the same cost, a multiscan monitor cannot be perfected (for all design criteria) as a simpler, single frequency design. However, a multiscan monitor is not inferior in any performance area; it depends on how manufacturers implement their topology. Indeed, there may be design areas where a multiscan monitor is inferior, or a multiscan may be the equal of a single frequency monitor. It just may cost more.

I selected the multiscan monitors reviewed here for their high resolution color capabilities and small dot pitch. They can all accept RGB analog signals at almost any commonly used horizontal scanning frequency. The horizontal scan frequency is one of several parameters that determines what appears on the CRT screen and what it looks like when it gets there. A higher horizontal frequency implies that more lines (left to right) are etched on the face of the CRT before retrace. Retrace brings the scanning electron beam back to the upper left hand corner and begins the entire sequence again.

To a lesser degree, these monitors also accept different vertical sync frequencies. Vertical sync or "scans" of less than 60 Hz can cause visible flicker. Faster scans might cause flicker, depending on ambient lighting. Our room lights actually blink on and off at 60 Hz intervals, so any significant differences from that rate can cause interference we pick up as flicker. Our eyes, brain, and body are accustomed to 60 Hz rates and fields, including the electricity in our homes, our lights, and from our computer screens. Any deviation (For instance, Europe uses 50 Hz) causes things to not look or feel right for a while. Finally, the persistence of the Phosphor within the CRT determines how long the painted image stays before it fades from view. All the monitors reviewed here contain P-22 (medium persistence phosphors), optimized for 1/60th second frame rates.

Frame rate is related to the interval of "painting" the CRT screen. If the vertical scan frequency is 60 Hz, the painting of the CRT is done 60 times per second. The current version of the Denise Chip in the Amiga 1000, 500, and 2000 gives us a about 200 scan lines in 1/60th of a second for each frame. Each scan line begins at the left and ends on the right side of the screen. Each line from the top of the screen image to the bottom

encompasses about 640 dots, and 200 scan lines in normal resolution mode. This set-up translates to an electrical horizontal scan frequency of 15.75 Khz. A horizontal scan of 31.5 Khz would give us about 400 scan lines.

If the Frame Rate is reduced to 1/30th of a second, we can produce 400 scan lines on the screen, but we can only paint it 30 times per second. This is how the current 640 x 400 line interlaced mode works. Each interlaced frame lasts twice as long on the CRT, but each alternate set of 200 lines fades away in 1/60th of a second. The effect is the well-known high res " flickering effect " between adjacent, dissimilar colored scan lines.

Commodore has indicated that in late 1988 or in 1989, a revised Denise chip will allow 400 scan lines within a 1/60th second frame rate. If the same number of colors and other capabilities are maintained, the video signal will require a non-interlaced analog monitor capable of accepting the resulting 31.5 Khz Horizontal Scan rate. Without any modifications, all multiscan monitors should be quite capable at that doubled scan frequency and the corresponding 640 x 400 screen resolution.

The Tests

When testing monitors, what you see is your primary concern; how the manufacturer gets the performance is secondary. Most often linear distortion and color fringing are the most noticeable problems. Linear distortion usually indicates a departure from absolute vertical linearity, denoted by lines that either bow inward ("pincushion" distortion), outward ("barrel" distortion), or some uneven combination. Any horizontal or vertical discrepancy is noticeable on the screen and can be a function of the circuitry, the CRT, or the CRT's shielding (or lack thereof).

Color fringing or "misconvergence" can appear anywhere on the screen. Since this misalignment is easily noticeable, it is significant. If the misconvergence is in only one area of the screen or near the edges, it indicates slightly more deflection for one of the electron guns. The result can easily be seen as white text with either a green, red, or blue color fringe.

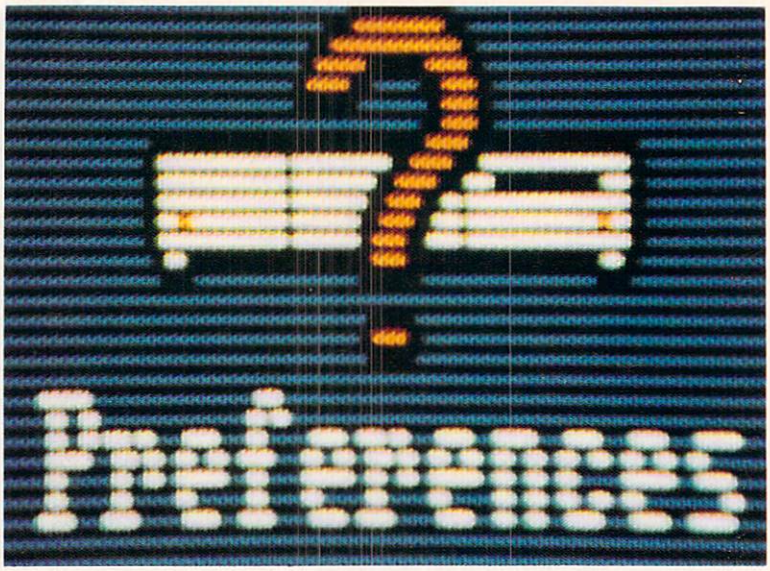
Other items that define a monitor's performance include color changes—usually affecting browns or oranges with changes in either brightness or contrast level; blooming—changing the size of the image with changes in brightness or contrast; video bandwidth—a frequency response rating for flat response to video signals (wider bandwidth usually means clearer definition of the screen image); and power consumption—tells how much electricity the electronics use.

(continued)

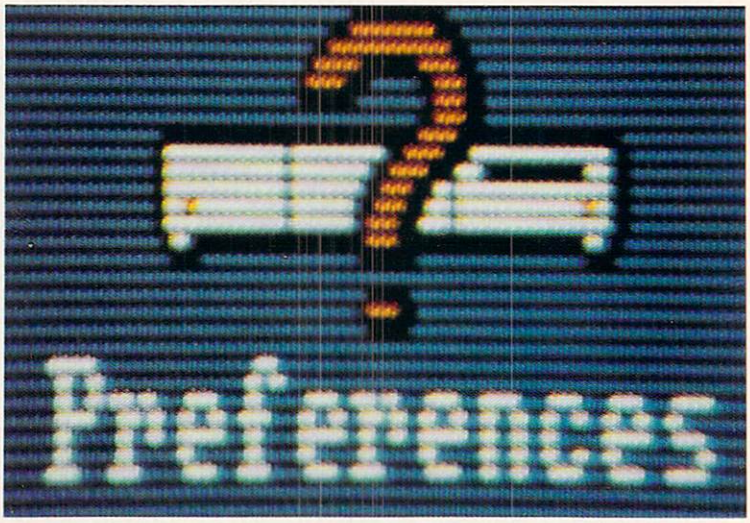
Details down to the pixel on the AMIGA A-1080 monitor.

Close up on the Mitsubishi AUM 1371A monitor.

Macro view of Preferences on the Logitech Autosync monitor.



Up close and personal with the Magnavox 8CM873 monitor.



A close look at Preferences on the Thomson 4375M.

An actual "in-use" test that I call "The Edge Jitter Test" is used here. This test reflects the stability of the left and right edges of a full screen when the center of the screen is updated very quickly. The test can be easily duplicated with Workbench 1.2's "boxes" demo zoomed to full screen size as the only task running.

This test is performed with the brightness control at maximum and the contrast varied from minimum to midpoint (using center detent if available) to maximum contrast. Under these conditions, the left and right screen edges either jitter noticeably or not at all. A ruler is placed against the screen, to get a range of the jitter over a 30-second interval for each of these three test conditions.

The Edge Jitter Test reflects the stability of the high voltage regulation within the monitor. Therefore, it is somewhat indicative of the overall electrical performance (independent of the CRT) and its shadow mask alignment. A monitor having no visible edge jitter indicates near-perfect internal design. The A-1080 displays very little jitter; some of the other monitors have much more noticeable edge jitter.

I'd suggest some caution against determining overall results from the Edge Jitter Test because, in normal use, the display is usually not updated so rapidly. Therefore, what is reflected in this test doesn't come into play much. However, when a screen is updated rapidly over most of its area, a jitter may correlate with "eye fatigue" or other visual disturbances. There are many vague VDT-related complaints for which there are no absolute tests or measurements. In some way, screen jitter is related to screen flicker, and the eye/brain sees it and reacts to it.

The Reviews

The Amiga A-1080 Monitor

A monitor's dot pitch determines the smallest point size on the screen. With its 0.42 mm dot pitch, the Amiga A-1080 is bested by all multiscan monitors reviewed here. The other units all have 0.31 mm dot pitch or better. The A-1080 excels in vertical linearity, though. Vertical lines are quite straight on the Amiga monitor, whether they are near the center, or towards the edge of the screen. Just as it should be. The Amiga monitor has a single side-mounted speaker that is reasonably good and fairly loud.

(continued)

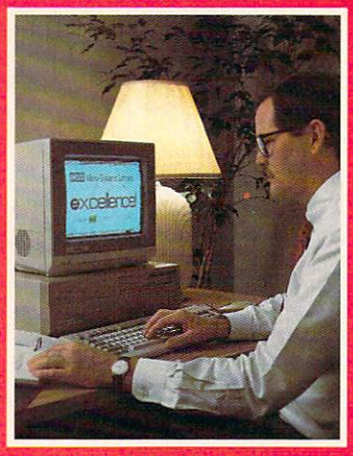
Amiga A-1080

Pros: Speaker & headphone jack; easy to hook to Amiga
Cons: Unsaturated colors; broad dot pitch
Linear Distortion: Above average
Color Fringing: Average
Color Changes: Orange changes to red as brightness reduces to minimum
Dot Pitch: 0.42 mm.
Blooming: Above average: image size is stable.
Video Bandwidth: Below average: approximately 15 Mhz.
Rated Resolution: Below average: 640 dots x 200 lines
Power Consumption: Low: about 75 Watts
Date of Manufacture: November 1985

Edge Jitter Test
Minimum Contrast: None
Midpoint Contrast: None
Maximum Contrast: Average up to 1/32"

Overall Rating: *Fairly good performance*

Upgrade from Perfect to Excellence!

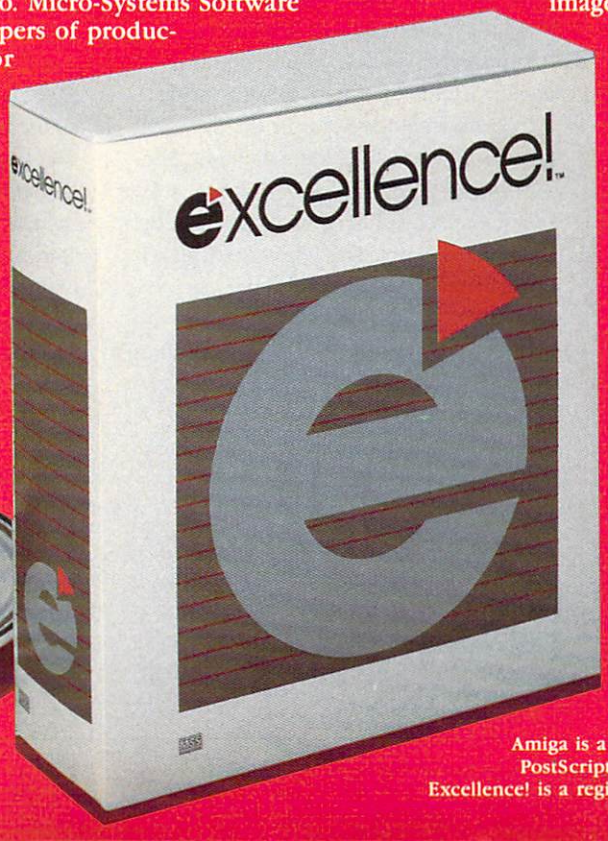


Micro-Systems Software is committed to a higher standard of excellence. And we're ready to prove it! Our newest Amiga product is a full-featured word processor that exemplifies our commitment to the Amiga. And to you. We have appropriately named it excellence! for obvious reasons. First of all, users of our popular word processor Scribble! told us about the features they wanted in a full-featured word processor. So we compiled their suggestions and designed excellence!, a program that sets new standards for word processing. And more importantly, excellence! has been developed specifically for the Amiga, on the Amiga. It takes advantage of the user-friendly Amiga interface and is designed to be intuitive in a way no other word processor can match. An important point: several companies, new to the Amiga market, want you to think their track record with other computer systems makes them instant experts with your Amiga. That just isn't so. Micro-Systems Software is one of the pioneer developers of productivity software exclusively for the Amiga! *We know your Amiga inside out.* So, features you once thought to be luxuries, you can now consider basics. Excellence! has all the powerful features required of a modern word processor, in a package sophisticated enough to use in desktop publishing.

There are always minor differences between programs designed for the same application. *Before* you make your choice, consider these major differences between excellence! and several well-known word processing programs! Excellence! processes words perfectly and does it faster than any other WYSIWYG word processing program available, giving the text-only programs a race for their money! (Not all programs claiming to be WYSIWYG really are. Excellence! shows you *everything*, including super- and sub-scripts, headers, footers, footnotes, colors, and graphics!) Excellence! has all of the standard features too, including: mail merge, full clipboard support, full text styling, multiple methods of movement within documents, search and replace, and printer control! Excellence! has a dramatic collection of features that place it at the forefront of a new generation of full-featured word processsors. Its innovations include: full color support of text, inclusion of IFF graphic images, spelling check as you type, basic math capabilities within documents, multiple column support, proportional font support, Index generator, Table of Contents generator, integrated Thesaurus, integrated Grammatical and Style checker, and PostScript output!

Features	excellence!	Word Perfect v4.1	Visa Write Desktop	Pro Write	Kind Words	Text Craft Plus
WYSIWYG (What You See Is What You Get)	✓		Partial	✓	Partial	✓
Grammatical & Style Checker	✓					
PostScript Output	✓					
Include Pictures With Text	✓		Limited	✓	✓	
Fully Clipboard Compatible	✓		✓	✓	✓	✓
Multiple Proportional Fonts	✓	Not Yet		✓		
Color Support	✓			✓		
Spelling Check As You Type	✓					
Math	✓	✓				
Multiple Columns	✓	✓				
Index Generator	✓	✓				
Table of Content Generator	✓	✓				
Thesaurus	✓	✓				
Limited Outliner		✓				
Mail Merge	✓	✓	✓		✓	✓

Excellence! is the perfect word processor for every need. From letters to newsletters; from the annual report to scientific research articles; from book reports to the book itself. If your writing aspires to excellence, you need a program to match!



MSS
Micro-Systems Software
Committed to excellence.

12798 West Forest Hill Boulevard
West Palm Beach, Florida 33414

Inquiries dial
1-(800) 327-8724
in Florida: (305) 790-0770

I prefer to use the Amiga monitor with the brightness at minimum and the contrast at maximum. This setting avoids the hazy, gray totally washed-out look that occurs if these settings are reversed. That washed-out look of unsaturated colors (with a grayish tinge to everything) from blues to blacks was a bit disconcerting to users. The A-1080 also has a lesser video bandwidth, since it was designed primarily for a resolution of 640 x 200. It often produces a haze over the colors and suffers a lack of color saturation. The A-1080 is a nice average monitor on an above average computer.

Magnavox 8CM873 Multi-Mode

It was the end of winter, and the UPS man was drenched from the rain. He brought a large package up the stairs. The package was a top-of-the-line Magnavox Multi-Mode monitor.

The Magnavox Multi-Mode has switch-selectable RGB digital and RGB analog inputs. The short users manual describes the Magnavox Multi-Mode as compatible with both the Amiga's RGB analog video and the IBM PC-XT's Digital CGA/EGA TTL Video outputs. While the Multi-Mode is compatible with most video adapters and standards, selection of the vertical scan frequency is not quite automatic.

The frequency scanning range is divided between three "V Size" controls. These screwdriver or alignment tool adjustments are recessed and accessible at the rear of the monitor. Adjusting them is quite a pain. Fortunately, they really only need to be adjusted once, unless you change computers or display standards. For optimum screen stability, one or more of these controls must be manually adjusted to the scanning frequency.

The pinout of the Magnavox DB-9 input jack appears exactly the same as the one used on the Commodore A-1080 RGB Monitor, so the Commodore cable can be used to connect the Multi-Mode to the Amiga. The color, size, and weight of the Multi-mode and Amiga A-1080 monitor are quite similar. The Magnavox occupies slightly more space, since the rear protrusion is wider but not deeper.

The CRT is billed as 14 inches diagonally, but my ruler says its viewing area is 13 inches. The "bulb" or CRT on the Magnavox is much darker than the A-1080's. The "dark bulb" on the Multi-Mode is indicative of its better contrast. Conversely, less is transmitted from within the CRT tube. This design results in blacker blacks and more vivid colors, but requires more power and generates more heat within the set.

The Magnavox works with the Amiga 1000, 500, or 2000 without anything beyond a matching cable. The two rear-mounted switches are the digital/analog and the normal/special. In analog mode, the normal/special switch has no effect. Aside from an RCA jack for audio, a DB-9 connector is the only input. The cable must be wired for one of the three compatible systems: CGA TTL Digital, EGA TTL Digital or RGB Analog. Magnavox sells a cable, or you can use the one that comes with the Amiga A-1080 monitor.

The Multi-Mode is packed with electronic circuitry, thus dissipating a great deal of heat. Several user controls are located under the front bezel. These controls set the horizontal direction, brightness, contrast, and volume. There are no center detents on any of these controls. You cannot set the color or tint in digital or analog mode. The left side has a side-mounted speaker and an earphone jack. Inside the set, some controls are sealed, but many are not. The convergence and other settings on the CRT's yolk are glued tight, so that the alignment problems might not be caused by shifting of the physical alignment, but rather by electromagnetic problems with the design (perhaps resulting partially from the speaker magnet).

With the contrast at about midpoint and the brightness upped by about a third, the blacks are quite black, and you get a brilliant display. Setting the contrast to maximum with the brightness fairly high causes the dots to get somewhat thicker and fatter (blur). Built-in limit circuits automatically reduce the brightness and preserve "black level" to some degree. They also reduce blooming effects. The Multi-Mode often seems too bright—it almost hurts the eyes!

Going back to the Amiga monitor is really quite a compromise, color and "lumens-wise." If the Multi-Mode only had the Amiga's pure vertical linearity (straight up and down lines), it would be just about perfect. At maximum, the sound isn't nearly as loud as the Amiga monitor's...but who cares! The color, the color, the color!

To get those vibrant, saturated colors at a high intensity level, the Magnavox CRT (made by Matsushita in our samples) requires a good deal of power behind it. Therefore, the circuitry must expend more energy, and the Multi-Mode gets quite hot, much hotter than the Amiga monitor, and hotter than any monitor tested here. The first 8CM873 tested, was eventually returned because of the tremendous internal heat generated. After a few hours, the top of the cabinet smelled like it was beginning to melt. At first, I assumed the intense heat to be a defect in the unit. After evaluating two more samples, I decided this was not a defect, but the monitor's normal operation.

The Multi-Mode also seems to have a significant linearity distortion problem. The first sample had several defects, including inward bending of vertical lines near the CRT's outer edges. This "pincushion" distortion was quite noticeable. When dragging a box across the Workbench screen, the white lines were straight near the center, but they arched inward significantly towards each side edge.

The first sample also suffered from very bad color fringing on the right side, top, and bottom. When that same box was on the far right of the screen, near the top white line turned pinkish, while near the bottom of the screen became greenish. A

bizarre flagging or bending of lines on the top most half inch of the screen towards the left also occurred.

All in all, the first unit was badly aligned and hardly what you would expect from a monitor with a list price of $900.00 and a street price of around $500.00. Therefore, I obtained a second sample of the 8CM873; it was much better than the first, but still not perfect. The second sample still suffered from pincushion distortion and had the flagging at the top of the screen (but less pronounced). Fringing was much less. In the second sample, with both the brightness and contrast turned down, the whites turned brownish. Moving the contrast control through its range changed the colors somewhat.

The third sample of the 8CM873 seemed (by a slight bit) the best of the three. All three units are from the same lot, manufactured in October, 1986. Perhaps Magnavox has improved its quality control since then. The third unit had some fringing (misconvergence), but only in the upper right hand corner. On the left edge, vertical lines were slightly bowed in; on the right edge, they were somewhat better. Away from the edge, things seemed quite linear. The center showed a very minor defect—an area slightly darker than the surrounding area.

When the unit was placed on the Amiga the defects were not quite as bad as in the previous two samples. Amazingly, when the Multi-Mode is sitting on top of the A-1080 monitor, it gets even better! The center zone of darkness disappears, and the top inch of the screen flagged in a smoother, more linear manner. Vertical lines at the edge still arched inward, but to a lesser degree.

Some of these defects (excluding misconvergence) were obviously related to stray magnetic fields. At least part of the problem is the Amiga! A little noticed item: the Amiga A-1080 monitor has a heavy gauge 9" x 12" steel plate on its bottom that most other monitors don't have. The straight lines and pure linearity of the A-1080 are a function of the monitor's isolation from the Amiga's magnetic fields. I suspect Commodore knew this, and added the steel plate, but did not mention it in the manual.

In any case, if you decide to buy the Magnavox Multi-Mode, a steel plate 12-14 inches square (tie it to ground—couldn't hurt) may help if you see linear distortion.

With the brightness near the half point for optimum display and minimum contrast, there was no edge jitter. With the contrast at midpoint, jitter was just noticeable. At maximum contrast, jitter was 1/32" to 1/16" in the center of the screen as the display was updated. This is *not* one of the worst showings encountered. Most of the time this type of performance is quite irrelevant, but it is noticeable during certain tasks.

Is the Multi-Mode a practical monitor for the Amiga? I guess the answer depends on whether or not you can ignore the pincushion distortion found on all three samples. If you can ignore it, the beauty of the colors makes the monitor practical. If absolute linearity is your preference, it is impractical. It's a shame that the purity of linearity doesn't approach the purity of the stunning colors.

What the Magnavox does right, it does very well. It displays brilliant colors and fine detail with its small dot pitch. The 8CM873 has a resolution of 926 dots x 580 lines; a resolution far beyond what the Amiga can now display. While we cannot use the resolution, we can use the color. The image is more vibrant than the same image on the Amiga monitor.

Newer Multi-Mode monitors reportedly have a non-glare surface on the CRT. (None of the samples I tested had this feature.) Perhaps it has been upgraded with a different CRT. Maybe Magnavox has improved shielding from internal and external fields. The unit was recently discontinued by NAT/Maganavox; try to test it on an Amiga before buying. The 8CM873 Multi-Mode may not be perfection, but it can be very good.

Magnavox Multi-Mode

Pros: Speaker & headphone jack; brilliant colors; easy to hook to Amiga
Cons: Much internal heat; vertical non-linearity
Linear Distortion: Combination of pincushion and barrel distortion
Color Fringing: Slightly below than average; varied with each sample
Color Changes: Orange changes to red as brightness is reduced to minimum

Dot Pitch: 0.31 mm.
Blooming: Small image size changes along with changes in brightness or contrast
Video Bandwidth: Below average: RATED: 25 Mhz.
Rated Resolution: Average: 926 dots x 580 lines
Horizontal frequency range: 15.5Khz-34Khz
Vertical frequency range: 50Hz-70Hz
Power Consumption: High: 130 Watts
Date of Manufacture: October, 1986

Edge Jitter Test
Minimum Contrast: None
Midpoint Contrast: Noticeable
Maximum Contrast: Avg 1/32" to 1/16"

Overall Rating: *Brilliant color with a few minor faults.*

The Mitsubishi AUM-1371A

The UPS man was now quite accustomed to delivering thirty pound packages. As I unpacked this monitor with its pitch black bulb, I had a feeling it was something special.

(continued)

The Mitsubishi Diamond Scan Monitor, model AUM-1371A, was replaced by the AUM-1381 in August. The models are essentially the same with one known improvement. In the newer unit, video modes are switched without the audible "click" that accompanies such changes in the 1371A unit. This improvement is not relevant to the Amiga, so we did not test the newer unit.

Unfortunately, the Diamond Scan unit was not designed to connect easily to the Amiga. This connection had never been attempted before...with good reason. Unlike the Magnavox Multi-Mode (which, in its manual, made a point of supporting the Amiga), the Mitsubishi Diamond Scan has its analog mode geared to the IBM PGA graphics cards. The rear panels of both the Amiga A-1080 and Magnavox 8CM873 contain only a Male DB-9 connector for video. The Diamond Scan has its digital video input on a Female DB-9, but the analog video input is separate, connected by a Female DB-25.

I unpacked the Diamond Scan Monitor and got out its thin, but verbose manual. I noted four pins on the analog video connector that could be readily used. I scanned the manual for more useful information since this was new, unexplored territory.

The Mitsubishi Diamond Scan's case is cream colored, similar to the Magnavox Multi-Mode. The rear of the Mitsubishi has a recessed panel covered with controls and jacks, including six small thumbscrew type controls. Just below these are the BNC video input jack and two switches. The controls cover vertical and horizontal size and position. The final two controls are for tint and color, only when you are using the composite (NTSC) video input (BNC jack). This spot feeds in video from your VCR or camera. On the left are sockets for the detachable AC cord, the DB-9 for digital (TTL) video, and the DB-25 for analog video.

The switches are another set of goodies. Monochrome/Normal allows you to choose between a green-type monochrome display or a color CGA/EGA/VGA display when using the Digital TTL inputs. Overscan/Underscan adjusts picture size in one fell swoop. The last switch has three positions labeled Video, TTL, and Analog. The analog input is good for IBM PGA/ GA display adapter standards. As we will see, a problem pops up when using the Amiga's analog video.

The Diamond Scan manual describes all features and inputs. The scan frequency is automatically selected by internal circuitry; no controls are provided for manual adjustment.

The Amiga's oddball DB-23 video connector is a minor problem, but now I knew I had to make a suitable female DB-23-to-male DB-25 video cable. I got out the soldering iron and solder, shielded cable, and various connectors, but soon realized that the Mitsubishi doesn't seem to have a pin for the composite sync signal used by the Amiga monitor. Uh Oh! There are separate horizontal and vertical sync lines and lots of strange input lines, but none seemed to be for composite sync.

I always thought the A-1080 used the Amiga's composite sync, and the separate horizontal and vertical sync lines were left unused. The pinout of the A-1080 cable proved my information to be correct. The A-1080 cable looked like this:

Amiga DB-23 Video Connector

Pins	3	4	5	10	16,17,18	19,20
Signals	R	G	B	COMP SYNC	GND	SYNC GND
Pins	3	4	5	7	1	2

A-1080 DB-9 Video Connector

The solution is to hook up the separate horizontal sync and vertical sync lines (The Amiga also has those outputs) then hope. So we have our three color video lines—red, green, and blue—the system ground, and the two new sync lines. I made up the cable, and the first try of the Mitsubishi cable looked like this: Since the Amiga was already set up, I removed the A-1080 monitor, and connected the new cable and the Diamond Scan. I set the rear switch for analog and turned the Diamond Scan on. Presto! The scrambled picture quickly became clear. Amaranth smiled and cheered; it was very clear that the connector worked. For a suitable screen, I tweaked the size and position controls on the back of the monitors. I also adjusted the brightness and contrast controls. I clicked open some icons and tried to run the Workbench demos.

All appeared to be fine. I decided, "Let's play Arkanoid." OK. After putting the disk into the internal drive, I did a Control/Amiga/Amiga. The screen blanked, became white, gray, white, and then went blank. I waited. Nothing.

OK, maybe a glitch scrambled the Kickstart. There was no UPS on the system at this time. I got the Kickstart disk and turned the power off and on. A few seconds later, a white screen appeared and then the friendly hand asked for Kickstart. I inserted Kickstart and the internal drive sprang into action. There were some color changes and again a blank screen. Nothing. Where was the friendly hand asking for the Workbench disk?

I tried again. I checked the wires and everything. I was puzzled. The monitor had worked; the screen had shown a perfect Workbench. It had all been there: the Mouse-Clock, the performance monitor, and the Workbench demos. I had double clicked on icons and they worked. I disconnected the Mitsubishi and hooked up

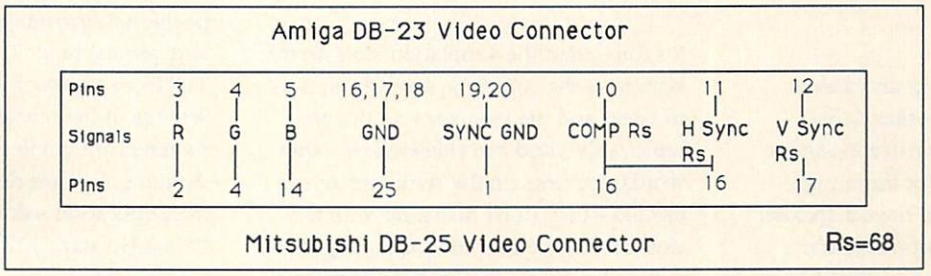
the Amiga A-1080. The Amiga powered up properly, asked for Kickstart and Workbench, and booted. No Problem.

Now I was confused. Hmmm. I pulled the DB-23 out and once again hooked up the Mitsubishi. Again, a clear, wonderful picture on the Diamond Scan. I did a Control/Amiga/Amiga. The screen blanked, became white, gray, white, and then went blank. Nothing again. Something was rotten in Denmark, and I aimed to find out what. I took the Amiga manual, the Diamond Scan manual, the original Commodore video cable, and started checking pins with a digital meter.

Now comes the interesting part. The Amiga manual defines pins 16,17,18,19, 20 as grounds. The pins are separated into two grounds on the A-1080's cable. I made them all one ground on the Mitsubishi cable, and I used the horizontal sync and vertical sync pins (11, 12 respectively), instead of the composite sync (10)! It worked fine..sort of. I separated the ground pins as the A-1080 cable had done, and made separate sync grounds and system grounds. To make a long story short, I tried this setup and it performed just like the previous cable. The monitor was fully functional after the computer booted up to the Workbench, but upon reboot, the system ended in a blank screen. Later, I tried various connections of the three Amiga sync lines (10, 11, 12—composite, horizontal, and vertical respectively).

At one point, while I was trying only one line by itself, the Diamond Scan sounded such a loud buzz, that I was startled. There is no speaker in this set; that noise came from its guts! I'm convinced the circuits which made that sound would surely produce smoke if left that way for any length of time. I eventually found the proper set of connections that produced a clear picture and let the Amiga boot properly.

In several conversations with Commodore Technical Support, I didn't find out if it was detrimental to parallel video pins 10 and 11.



This cable design shorts the composite sync and horizontal sync signals on the Amiga's video output. This could lead to eventual failure of the video circuitry in the computer, but it appeared to work perfectly. The results are unpredictable because the H-Sync line is normally unused with the A-1080 monitor. That caveat should be remembered—it could lead to a circuitry failure in the Amiga. I connected it and left it operating like this for several hours without any problems. This boot problem is related to some strange sensing on the composite sync line that occurs whenever the Amiga is cold booted or reset with CTRL/Amiga/Amiga. This sensing appears to be related to the Amiga's genlocking capability.

Later on, I discovered that the wiring diagram shown above also functions properly, but isolates the signal on pin 10 from the signal on pin 11. I tried this after the Commodore Amiga Developers suggested that each sync line (pins 10, 11, 12) have small 1/4 watt, 47 to 470 ohm resistor (Rs) in series:

Now that it works, how does it perform? Beautifully. The CRT is just slightly larger than the one on the A-1080. It has a very dark, somewhat purple, tint to the glass surface. This dark bulb, that Mitsubishi calls a "Diamond Matte Coating, Super-High Contrast Glass," gives a deep, dark black and a non-glare surface. This monitor has the blackest black of any that I have seen, allowing high contrast. I suspect it also limits transmission from within the tube, though. At settings of maximum brightness and maximum contrast, there is no blur, loss of clarity, or haze.

Overall the Diamond Scan is very linear with just the slightest touch of inward bowing of straight lines from the sides. The color fringing (misconvergence) is on a par with, or better than, the A-1080. Our sample gave just a touch of pink on the top right side and some minor fringing on the lowest scan lines. The fringing tended to be blue on the left side of center and red on the right.

The colors are medium bright and well saturated. When turned up to maximum brightness (which maintains a black background), the A-1080 and Magnavox Multi-Mode monitors get somewhat brighter. I prefer to watch the Diamond Scan with the brightness set at midpoint (Yes, there is a center detent), and the contrast set at maximum or just below.

There were times, in a brightly lit room, when I wished I could turn the contrast control another 20%. It is so clear and saturated that I longed for a few extra points. While brightness levels tend to vary with the age and usage of the CRT (and from unit to unit within a given model), I certainly found the display adequate, even intense. Incidentally, the brightness and contrast controls are mislabeled. I'm referring to them as they are labeled, not as they actually function. The Diamond Scan gets warm, not as hot as the Magnavox, but hotter than the Amiga A-1080. It uses about 50% more power than the A-1080 and about 30% less than the Magnavox Multi-Mode.

(continued)

As mentioned, the Edge Jitter Test measures the stability of the left and right edges of a full screen as it is rapidly updated. With the brightness at maximum and minimum contrast, the Diamond Scan had no jitter. With the contrast half way up, jitter was noticeable to about 1/32". At maximum contrast, jitter went to 1/16", and the entire screen bounced slightly—pretty poor stability. I used the Workbench 1.2 "Boxes" demo running at full speed.

The Diamond Scan lacks an internal audio amplifier and speaker. Most speakers in rather expensive monitors are mediocre at best. For the Amiga, however, it hurts not to have a speaker at all. As a high resolution text and graphics monitor, the AUM-1371A is quite outstanding. It's a good idea for you to have external amps and speakers. Remember the caveat, though: using your own video cable can be a problem.

The Logitech AutoSync Monitor

Upon opening the substantial white box, I noticed something different. This one has two main wires; the one for the A.C. power detaches, but the other one that carries the video doesn't detach ! Will this ever work? Quickly, I starting thumbing through the manual. Hmmm. Lots of pinouts and switches. I decided this versatile unit needed an adapter cable.

I also noted that the Logitech came with an attached tilt and swivel stand (albeit a small one) mounted flush with the bottom of the monitor. I tried to remove it, but it didn't want to be removed. Oh well.

It's funny that the ventilation slots on the bottom of the AutoSync travel from front to back, and the vent slots on the non-removable stand run sideways. In other words, the slots on the stand are almost useless—they don't match up with the slots on the monitor. Bad idea, Logitech. Still, at least the top of the stand is small and negates only a fraction of the monitors' bottom vent slots. The stand I used on the other monitors actually covered the entire bottoms. Both those monitors have feet, though, which raise them above the stand by 1/8th to 1/4 of an inch and allow some air to get in.

When the AutoSync's case rests parallel to the desk or table, the bezel is aimed up at a slight angle. The rear panel contains two main switches and five smaller DIP switches. The two main switches are TTL/ANALOG and SCAN MODES: Auto/Manual. The DIP switches control text color and include color switches for setting screen defaults in TTL Mode.

On the right side, near the front, is a little door. Behind that door are two more switches and six small knurled controls for adjusting brightness, contrast, vertical position, horizontal position, vertical size, and vertical hold. The first switch is TEXT on/off which selects whether the settings of the rear-mounted DIP switches are in effect or whether the default colors are determined solely by the application software (TTL Mode) for CGA/EGA standards.

Further reading of the manual revealed that when TEXT is ON, there is only one color on the screen ("Green," " Cyan, " etc). The second switch (H-Width) is more like an Overscan/Underscan switch, but it is effective only at higher scan modes.

The thirty-page AutoSync users' manual is one of the best monitor manuals I've seen. The horizontal scan frequency is automatically selected by internal circuitry. According to the manual, the vertical frequency scan is adjustable manually with the V-HOLD control.

I started to make a simple cable adapter by connecting the three RGB signal lines, the sync, and the ground to a female DB-9 connector. This connection went to a female DB-23 for the Amiga and was then mated with the male DB-9 permanently attached to the three-foot umbilical cord coming from the AutoSync. This adapter has lines one through three as red, green and blue, respectively. Pin four is a composite sync, and lines five through nine are grounded.

I set the rear switch for analog and depressed the ON switch under the front. A few seconds later a scrambled picture appeared and quickly became clear. I made a few minor adjustments on size and position. Not Bad. Now the test—

MITSUBISHI DIAMOND SCAN

Pros: Blackest blacks; saturated colors; very linear; better than average non-glare screen
Cons: Lacks speaker and headphone jack; difficult to hook to Amiga
Linear Distortion: Above average: very slight "pincushion"
Color Fringing: Above average
Color Changes: Orange changes to dark orange as brightness is reduced to minimum
Dot Pitch: 0.31 mm.
Blooming: Image size changes slightly with changes in brightness or contrast
Horizontal frequency range: 15.6Khz-35Khz
Vertical frequency range: 45Hz-75Hz
Video Bandwidth: Avg: RATED: 30 Mhz.
Rated Resolution: Average: 800 dots x 560 lines,
Power Consumption: Average: 90 Watts
Date of Manufacture: November 1986

The Edge Jitter Test

Minimum Contrast: None
Midpoint Contrast: Average 1/32"
Maximum Contrast: Below Average 1/16"

Overall Rating: *Nearly excellent performance*

(continued)

EXPLORE THE NEW WORLD

Artificial Intelligence
Expert Systems

You've heard the words. You probably know a bit about what they mean, too. You might associate them with white coats, Russian accents and shiny silver robots. What you might not know is that you can explore AI on your Amiga.

MAGELLAN™ by **Emerald Intelligence** offers performance and interface features UNAVAILABLE on the IBM-PC*, PS/2* or Apple Macintosh*. It was designed, developed and tested on the AMIGA for the AMIGA.

MAGELLAN turns *your* AMIGA into a powerful knowledge engineering workstation. It simplifies some of the complexities of artificial intelligence by using the power of the AMIGA. It is advanced enough to be simple.

The AMIGA's unique capabilities to support inexpensive desktop video, desktop publishing, audio/voice synthesis, extended memory, networking and high-resolution graphics make it ideal for advanced artificial intelligence development. Advanced to the state of being useful, not a graduate-school curiosity. Advanced enough to interface with the real world, and get to work. Advanced enough to be useful.

Use **MAGELLAN** to capture the expertise of your best salesperson, to help junior salespeople "learn the ropes". Build a system to diagnose sick AMIGA's and prescribe repair procedures. Weigh purchasing options in your job or at home. Evaluate performance of the hockey team. Drive animations with rules of behavior of the "actors" or objects. Create original computer art and music. Add real artificial intelligence to games to take them to the edge.

Best of all, use **MAGELLAN** to write programs. No BASIC, no "C", no compiler, no linking, no waiting. No kidding. With programs written with rules, changes can be made interactively — like BASIC — but much, much more efficiently. People don't think in flow charts, or "C" language structures or little curly brackets or parentheses. People think in terms of "IF — THEN". As a child your mother would say "If you take that cookie, then I'll . . ." Rules like this are how people work, how people decide. This is how computers will be programmed by our children.

Welcome artificial intelligence to the real world. Explore **MAGELLAN.**

M A G E L L A N

List Price
$195

Emerald
Intelligence

334 South State Street, Ann Arbor, Michigan 48104. (313) 663-8757.

*IBM-PC, PS/2 are registered trademarks of International Business Machines, Inc. *Apple Macintosh is a registered trademark of Apple Computer Corp. *AMIGA is a registered trademark of Commodore Business Machines, Ltd.

Control/Amiga/Amiga? Everything works just fine.

The AutoSync performs very nicely with the Amiga. The screen has a very dark, non-glare surface. It is not exactly like the CRT in the Mitsubishi Diamond Scan; it's surface is lighter in color, and it reflects more light. It is, however, the same CRT as NEC uses in their multisync. The background is quite black and contrast can be very good. Vertical linearity here was the best of the four tested multiscan monitors. Overall, the display was excellent. However, with the contrast set near maximum, everything became blurred. At minimum contrast, there was very little contrast. At most common settings, there was no blur.

Varying the brightness from minimum to maximum caused no color change in the orange. Brightness control ranged from total darkness to bright. Like the Diamond Scan, it didn't get very "bright," even at maximum brightness and maximum contrast. Actually, the screen provides exceptional clarity if both these controls are set just below maximum. The AutoSync's performs quite impressively in analog mode. It is possible to set Preferences so the Workbench screen moves up too high. Under these conditions, severe flagging occurs. Text at the top of the screen breaks up and becomes unreadable. It is unlikely that anyone would try to use it this way, though.

With Workbench centered on the CRT, the AutoSync is very linear. In fact, vertical and horizontal linearity seem to be its best features. Misconvergence wasn't very apparent, but the colors weren't terribly saturated and the brightness didn't go all that bright without blur. While the rear of the Diamond Scan got quite warm and the Magnavox got quite hot, the AutoSync is also quite warm running in analog mode. It dissipates less heat than the Magnavox Multi-Mode, and is on a par with the Diamond Scan. Perhaps it is just ventilated better.

Amiga DB-23 Video Connector

Pins	3	4	5	10	16,17,18,19,20
Signals	R	G	B	COMP SYNC	GND
Pins	1	2	3	4	5,6,7,8,9

AutoSync DB-9 Video Connector

At maximum brightness and contrast, the AutoSync was up to one 1/32" on the Edge Jitter Test (mostly in the center of the screen). This same effect was much less apparent on the Mitsubishi, and perhaps even less than on the Magnavox. At maximum brightness and half way up on the contrast (no center detents), jitter was very slight. The AutoSync lacks an internal audio amplifier and speaker. As a high resolution text and graphics monitor, the AutoSync is A-O.K. It is a very good choice for the Amiga if you have external amps and speakers.

The Thomson UltraScan Monitor

When I opened the 4375M box and saw this monitor, I had a feeling of deja-vu. I knew I had seen it all before. Except for the faceplate, the Thomson was the Mitsubishi Diamond Scan all over again (albeit with a few basic changes.) The rear panel has an RCA jack for video input in place of the BNC jack. The Thomson bezel is darker colored and differently shaped. The power switch on the Mitsubishi version is flush with the front bezel surface; the Thomson power switch button is smaller and sticks out slightly. The control labeled "brightness" has a center detent on both units.

After testing, I can state that the Diamond Scan and UltraScan are one and the same. They have to be—they even have the same FCC ID Number. Do they work the same? Pretty much. The CRT is the same: jet black bulb and a non-glare surface. Text and graphics are clean and clear. Overall, they are as close as two peas in a pod.

Logitech AutoSync

Pros: Good blacks; wide range of color contrast; average non-glare screen
Cons: Lacks speaker and headphone jack; difficult to hook to Amiga; colors not very saturated; gets blurred near maximum brightness or contrast level
Linear Distortion: Above average: none
Color Fringing: Above average
Dot Pitch: 0.31 mm.
Blooming: Image size doesn't change significantly with changes in brightness or contrast
Horizontal frequency range: 15.5Khz-35Khz
Vertical frequency range: 45Hz-80Hz
Video Bandwidth: Well above average: RATED: 40 Mhz.
Rated Resolution: Average: 800 dots x 560 lines
Power Consumption: Average: 100 Watts
Date of Manufacture: January 1988

The Edge Jitter Test
Minimum Contrast: None
Midpoint Contrast: Slight
Maximum Contrast: Average 1/32"

Overall Rating: *Certainly well above average.*

I noted two unit-to-unit differences. When the Mitsubishi is first turned on, the screen displays a white with a red, pinkish hue. This gets progressively whiter over a period of ninety seconds or so. The Thomson comes up with a whiter white, meaning all three electron guns are better matched at turn-on. After several minutes, you can't tell them apart over most of the screen. Both units show some minor fringing effects, mostly at the very bottom of the screen. The Mitsubishi tends toward a predominantly blue fringe on the left side of center and a red fringe on the right. The Thomson has a predominantly green fringe on the right side.

With the brightness at maximum and contrast at minimum, the Thomson had no jitter. With the contrast half way up, jitter was very slight. At maximum contrast, jitter became significant (about 1/16"), and the entire screen bounced slightly. Like the Mitsubishi, the Thomson also exhibited pretty poor stability.

The UtraSync unit has recently been discontinued, but it still has a significant presence in the retail channels. It costs about $100 less than the Mitsubishi Diamond Scan.

THOMSON ULTRASYNC

Pros: Blackest blacks; saturated colors; better than average non-glare screen.
Cons: Lacks speaker and headphone jack; difficult to hook to Amiga.
Linear Distortion: Average; slight pincushion.
Color Fringing: Above average
Color Changes: Orange changes to dark orange as brightness is reduced to minimum
Dot Pitch: 0.31 mm.
Blooming: Image size changes slightly with changes in brightness or contrast
Video Bandwidth: Average; RATED: 30 Mhz.
Rated Resolution: Average; 800 dots x 560 lines
Power Consumption: Average; ~ 90 Watts
Date of Manufacture: December 1986

The Edge Jitter Test
Minimum Contrast: None
Midpoint Contrast: Very slight
Maximum Contrast: Below Average 1/16"

Overall Rating: *Slightly worse than the Mitsubishi in some areas; better in others. Essentially the same.*

Special thanks to Jim Black of BCD for the DB-23, DB-9, and DIN connectors used to make the video cables.

•AC•

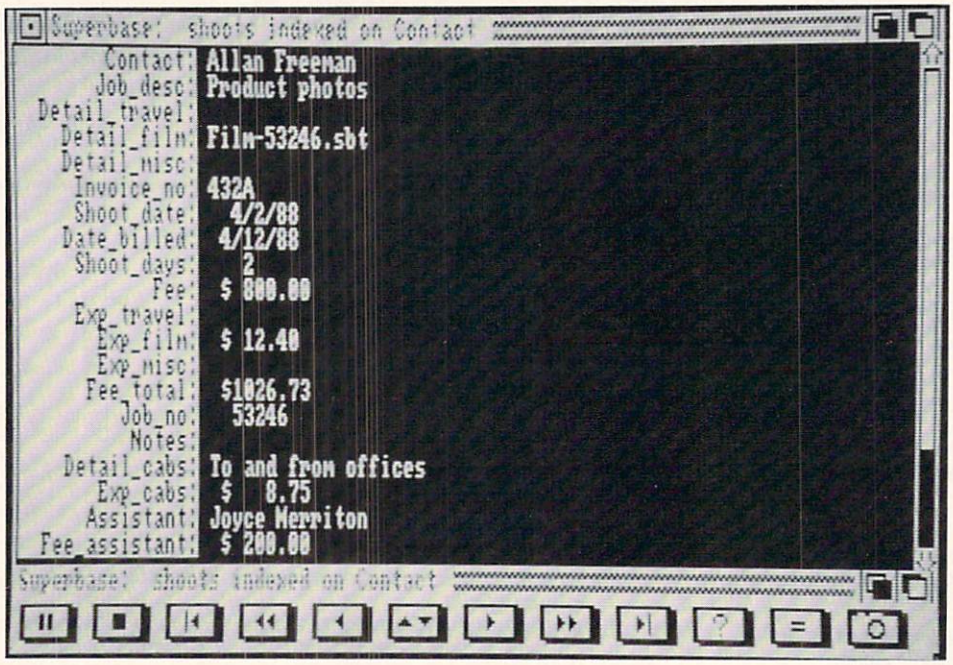
Record-Keeping for Freelancers

Using Superbase Professional, create a simple record-keeping system for journalistic photographers.

by Marion Deland

This Superbase Professional application is a simple record-keeping system for journalistic photographers. With some minor adaptations, it would serve most freelancers, especially those in some form of graphic arts.

First, a word about the program. Superbase Professional, created by Precision Software in England, and marketed in the U.S. by Progressive Peripherals & Software, is the kind of software we want to see for the Amiga. It is a powerful, easy to use professional database management system.



It makes full use of the Amiga environment—windows, menus, multitasking—to give both the beginner and the sophisticated database programmer the means to develop multi-file applications, complete with customized screens, menus and requestors.

If you've used Superbase Personal, you'll feel at home with SBPro. It includes the database/query setup of Superbase Personal—VCR-style controls and all—with enhancements. Added are a text editor, a forms editor and, of course, a programming language.

There is one negative point. Under pressure to bring the program to market, Precision Software evidently released it without spending enough time on gamma testing. I have already been issued an update (Version 2.02), and there are still occasional bugs. Now that many of us have expanded memory and are in the habit of multitasking, a visit from the guru that takes three programs and the RAM disk with it is no joke. On the other hand, far less complex programs have been released with many more problems.

Enough about Superbase Professional—let's get on with the application.

Programs this powerful can be intimidating, so this application is part tutorial. It will give you a look at each feature of SBPro and some tips in using them.

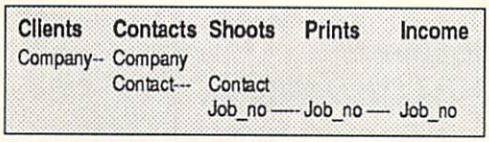
First, we'll create the files and some dummy records. Then we'll do a report, using the QUERY function to link files and extrapolate their data. Next, we'll use the text editor to write a MAIL MERGE letter. And, last but not least, we'll create a multi-file invoice, using the forms editor.

Ready? Off we go, then.

Creating the Files

There are five files. (They're fairly small!) Remember, one of the advantages of a relational database is that you can keep information in several small, related files rather than repeating the same material over and over in large files.

The files are linked through fields with common information, like this:



These linking fields are also indexed fields. (Indexes are the way SBPro keeps track of your data.) Only indexed fields are recognized by the Forms Editor as possible linking fields.

CLIENTS

The first file you will need is "Clients."

Use the PROJECT menu to create a NEW FILE, and set up these fields. (Manual 1 will tell you more about how to do this.)

For this file, ignore the "multiple response" box; each field has one response. The ZIP_code is a text field, rather than a numeric one, to allow for the hyphen in the new nine-digit codes. Notice there are no spaces in the field names.

DATE FIELD (CALCULATED)

Date_entered [Format = mm/dd/yy]
[Calculation = "TODAY"]

(About this calculation—there is an error in the manual. Use "TODAY", not "DATE".)

If you like, you can validate this date by entering "Date_entered <= TODAY " in the validation requestor.

When you've finished creating the fields, click on "OK" and Superbase will ask you to create an indexed field. Make "Company" a unique index, then save the file.

Use the SYSTEM STATUS FILE menu to see the file definition. It will give you some statistics, then list the fields. Notice the two columns at the far right. These contain the FORM VIEW coordinates for the fields. If you have experimented with FORM VIEW by repositioning the fields, the coordinates will be different from those listed in Table One.

Table One:
CLIENTS File Definition

```
Company      ;TXT          REQ IXU  ;30      ;0    ;0    ;
Street       ;TXT          REQ      ;30      ;1    ;0    ;
City         ;TXT          REQ      ;20      ;2    ;0    ;
State        ;TXT          REQ      ;20U     ;3    ;0    ;
ZIP_code     ;TXT                   ;10      ;4    ;0    ;
Date_entered ;DAT COV RDO           ;mm/dd/yy ;5   ;0    ;>>
TODAY
Date_entered <= TODAY
```

CONTACTS

"Contacts" is a separate file; you may have several different contacts at the same company. If this isn't the case, you may prefer to combine "Clients" and "Contacts" into one file. If, however, you work with agencies and their clients, another text field—"Agency"—could be added.

TEXT FIELDS

Field Name	Size	Required Field?
Company	20	Yes
Contact	30	Yes
Salutation	20	Yes
Title	40	No
Telephone	12	No

For the "Telephone" field, type "2" in the box for multiple responses. Your contact may have more than one phone number.

Notice the telephone number is a text field rather than a numeric, so you can include hyphens, brackets, etc.

What's a salutation? It's the form of address you use in a letter. In a letter to Mary Jones, for example, you might say "Dear Ms. Jones," "Dear Mary," or "Senator Jones:".

Table Two:
CONTACTS File Definition

```
Company      ;TXT          IXD ;20   ;0    ;0    ;
Contact      ;TXT          IXU ;30   ;1    ;0    ;
Salutation   ;TXT          ;20       ;2    ;0    ;
Title        ;TXT          ;40       ;3    ;0    ;
Telephone    ;TXT          ;12 M2    ;4    ;0    ;
Notes        ;EXT          ;20       ;6    ;0    ;
Date_entered ;DAT COV RDO  ;dd/mm/yy ;7    ;0    ;>>
TODAY
Date_entered.Contacts <= TODAY
```

EXTERNAL TEXT FIELD

This file includes an external text field. Instead of text, the field will contain the name of an external file, like "NotesJones.sbt".

Field Name	Size
Notes	20

DATE FIELD

Date_entered [same as "client" file]

(continued)

Make "Company" a normal index and "Contact" a unique index. Notice there is a "Company" field in both files; it will form the link between them.

While it is easier to see joins between files if you give the linking fields the same name in both files, it is not necessary. Only the contents must be the same.

Later, if you get into programming SBPro, you may want to rename one of these fields; otherwise you have to use the file extension (Company.Clients or Company.Contacts) each time you refer to them.

The file definition will look like Table Two.

SHOOTS

The next file is "Shoots" (photography assignments). It includes calculations of fees and a sequential job number.

TEXT FIELDS

Field Name	Size	Required Field?
Contact	20	Yes
Job_desc	40	No
Detail_travel	30	No
Detail_cabs	30	No
PO_number	20	No
Invoice_no	6	No
Assistant	30	No

DATE FIELDS

Shoot_date [format: mm/dd/yy]
Date_billed

NUMERIC FIELDS

Shoot_days [format: -99.]
[number of days involved in shooting job]
Exp_travel [format: -$ 9999.00]
[travel expenses if reimbursable]
Exp_film [format: -$ 999.00]
[cost of film if reimbursable]
Exp_misc. [format: -$999.00]
[miscellaneous reimbursable expenses]
Exp_cabs
Fee_assistant
Phone_calls
Messengers
Meals

CONSTANT FIELD

Fee [Shoot_days * 500]

This formula will calculate your fee based on your daily rate (in this case $500). You can change the total when you enter the record—you may have different fees for different types of clients.

CALCULATED FIELDS

Fee_total
[Fee + Exp_travel + Exp_film + Exp_misc]
Job_no [53241 + (SER ("Shoots"))]

SBPro includes a SER (serial) function to keep track of the number of records that have been created. It is used here to update the job number.

When you seriously begin working with your database, enter into the "Job_no" calculation your last job number from your own records (53241, for example), and Superbase Professional will update the job number from that point. Note that this serial formula will result in a numeric job number only. String functions can be used with SER to create an alphanumeric job number.

EXTERNAL FIELDS

Field Name	size
Notes	20
[See "Notes.contacts" above.]	
Detail_film	20
Detail_misc	20

Make "Contact" a normal index and "Invoice_no" a unique index. The finished file definition will look like Table Three.

PRINTS

The "Prints" file is for recording print orders resulting from your shoots. It includes calculations of totals plus sales tax, and is linked to the "Shoots" file through the "Job_no" field.

TEXT FIELDS

Field Name	Size	Required Field?
Invoice_no	10	Yes
Project_type	10	No
[Is the job for a newsletter, annual report, etc.?]		
PO_Number	10	No

NUMERIC FIELDS

Amount	[format: -$ 9999.00]
Job_no	[format: -999999.]
No_sent	[format: -999.]
Messengers	[format: -$ 999.00]
Retouching	[format: -$ 9999.00]

Table Three: SHOOTS File Definition

Contact	;TXT	REQ IXD ;20	;0	;0	;
Job_desc	;TXT	;40	;1	;0	;
Detail_travel	;TXT	;30	;2	;0	;
Detail_cabs	;TXT	;30	;3	;0	;
PO_number	;TXT	;20	;4	;0	;
Invoice_no	;TXT	IXU ;6	;5	;0	;
Assistant	;TXT	;30	;6	;0	;
Shoot_date	;DAT CON	;mm/dd/yy	;8	;0	>
TODAY					
Date_billed	;DAT	;mm/dd/yy	;9	;0	;
Shoot_days	;NUM	;-999.	;10	;0	;
Exp_travel	;NUM	;-$ 9999.00	;11	;0	;
Exp_film	;NUM	;-$ 999.00	;12	;0	;
Exp_misc	;NUM	;-$ 9999.00	;13	;0	;
Exp_cabs	;NUM	;-$ 9999.00	;14	;0	;
Fee_assistant	;NUM	;-$ 9999.00	;15	;0	;
Phone_calls	;NUM	;-999.00	;16	;0	;
Messengers	;NUM	;-$ 999.00	;17	;0	;
Meals	;NUM	;-$ 999.00	;18	;0	;
Fee	;NUM CON	;-$ 9999.00	;19	;0	>
Shoot_days * 500					
Fee_total	;NUM CLC RDO	;-$ 9999.00	;20	;0	>
Fee + Exp_travel + Exp_film + Exp_misc					
Job_no	;NUM CLC RDO	;999999.	;21	;0	>
53241 + (SER ("Shoots"))					
Notes	;EXT	;20	;22	;0	;
Detail_film	;EXT	;20	;23	;0	;
Detail_misc	;EXT	;20	;24	;0	;

DATE FIELDS

Date_sent	[format: mm/dd/yy]
Date_invoiced	[format: mm/dd/yy]

EXTERNAL FIELDS

Print_details
Sample_photos

CALCULATION FIELDS

Field Name	Calculation
Sales_tax	Amount * [sales tax for your area— e.g. 8.25] /100
Sub_total	[Amount + Messengers.Prints + Retouching]
Print_total	[Sub_total + Sales_tax]

Make "Job_no" a normal index (you may have more than one print order from this shooting).

The last file is "Income," for recording the money that comes pouring in. It is linked to the "Shoots" file and the "Prints" file through the "Job_no" field.

DATE FIELD

Date_paid	[format: mm/dd/yy]

TEXT FIELD

Field Name	Size	Req Field?
Invoice_no [multiple: 3]	10	Yes

NUMERIC FIELDS

Amt_paid	[format: -$ 9999.00]
Job_no	[format: -999999.]

This time the index is "Job_no"—a normal index, because you might receive several payments for one job.

REPORTING YOUR DATA

Now that we've created the files, what can we do with them? First, let's enter some examples so we have something to work with.

You shot a job on March 12, 1984, for Mary Jones, Manager, Public Relations, of XYZ Corp., 57 Broadway, New York, NY 10025 (Phone 212-555-1116). The job took one day, and involved photographing chemists in the lab of the company's Stamford CT facility. The company will reimburse your expenses for train ($5.50) and taxi fares ($12.00) to Stamford, and for two rolls of color film.

The client ordered 4 prints—#s 12A and 15A from sheet 1 and #s 6 and 15 from sheet 2. You'll want to make a note in the "Contacts" file that Mary Jones pays bills promptly if they are directed to her personally. And the bill was indeed paid in full promptly on April 8, 1984—it's a test case, okay?

Open each file in turn, select NEW from the RECORD menu, fill in the information, and click twice to save the record.

When you reach the "Notes" field of the "Contacts" file, select PROJECT TEXT, type your note, save it as "JonesM", close the text editor, then enter "JonesM.sbt" in the "Notes" field. Use the same method to list the prints ordered in an external file for the "Print_details" field. (Use the job number as the name of the external file.)

Set up a few more sample jobs, then we'll develop a report that calculates your total billings.

Superbase Personal uses the QUERY to output your data. SBPro, on the other hand, gives you two options—the QUERY, and the REPORT feature of the forms editor.

The QUERY is quick and easy to set up, and in many cases it is all you will need. A REPORT form takes longer to set up, but it has more flexibility.

If you've used Superbase Personal, you'll find the QUERY has been slightly enhanced. There's a SAY option to let your Amiga speak the results, and loading a query will automatically load its associated files. You can also vary the text style attributes and foreground/background colors.

Select QUERY from the PROCESS menu and begin to build the query.

(continued)

First, the fields. The manual will tell you how to select them by using the length and position buttons to get a clear screen layout. These are the fields you will need:

```
&15Contact.Contacts
&20Company.Contacts
&10Fee_total AS "Shooting"      [From "Shoots"]
&12Print_total AS "Prints"      [From "Prints"]
&10Sales_tax                    [From "Prints:]
&10Print_total + Fee_total + Sales_tax AS
"Amount billed"
```

Next, you'll want the report to add up your billings—by client and in total. Click on the REPORT section and enter these.

REPORT SUM

```
Fee_total,Print_total,Sales_tax,"Amount billed"
GROUP Contact.Contacts SUM
Fee_total,Print_total, Sales_tax,"Amount
billed"
```

The FILTER box is where you create the links between files:

```
Job_no.prints = Job_no.Shoots AND
Contact.Shoots =
Contact.Contacts
```

Last but not least is the order. For our example, it doesn't much matter—we've entered only a few records—but in a real report you will want all jobs for the same person grouped together. So:

```
Contact.Shoots ASCENDING
```

All set? Click on "OK" and watch your report appear on the screen.

You can develop a more detailed report with the forms editor. Report forms can include BEFORE and AFTER REPORT/GROUP functions, especially useful for identifying the items in a SUMMARIZED report.

(Note: My copy had a bug—the SUMMARIZE keyword appeared on a line of its own in the finished report program, producing a syntax error. I moved it to read REPORT SUMMARIZE, and all was well.)

TEXT EDITOR

So far, we've dealt mostly with features that are included in both Superbase Personal and Superbase Professional—the database itself and the report feature. But SBPro also includes a text editor, a forms editor, and its own programming language, making it a very powerful program indeed.

The text editor is very easy to use. It's not meant to take the place of a word processor—just to produce simple text files that can be used as form letters for

mail merge operations or as external text files attached to records.

Let's create a form letter reminding inactive clients—those we haven't worked with for, say, a year—that we exist. (Instructions for this are in the first manual under PROCESS MAIL MERGE, pp. 5-17.)

Select PROJECT TEXT to open the text editor window. You might want to resize the window and change the margins (the RULER setting). Fields or date functions that you want to merge into your form letter are delimited by the ampersand (&), as follows:

&today&
&Company&
&Street&
&City& &State& &ZIP_code&
&Contact&

&Salutation&
It's been some time since we worked together. I have been doing some very interesting work in your field. I'd like to meet with you to show you my book, and see if there is anything I can help you with in the future.

I'll call for an appointment—I look forward to talking with you again.

Yours sincerely,

SAVE the text file.

Next, you need a single file containing the names and addresses you will need. Use the QUERY function to put this together. You will need three files—"Clients," "Contacts," and "Shoots." In addition to linking the three files through Company and Contact, you will need this filter:

AND Shoot_date < (TODAY-365)

Output the results to a new file, then open that file. Follow the manual's MAIL MERGE instructions to see the results.

FORMS EDITOR

The forms editor adds a new dimension to SBPro. You can create forms for data entry, billing, reports, and all kinds of business paperwork.

We're going to set up a form to print out invoices. The form will draw its information from four files and figure out the arithmetic.

First load the forms editor, either by PROJECT EDIT FORM from SBPro (if you have enough memory for both programs), or by closing down SBPro and clicking on the FormsEd icon (on its own disk). You will see the forms editor has its own menus. Files you have opened in SBPro must be opened separately in the forms editor, so use the PROJECT menu to open "Clients," "Contacts," "Shoots," and "Prints."

Start by entering the fields. Keep them simple—no "auto field names" or "auto field borders." For this example, you'll enter the text, boxes, rules, etc. separately.

(To line everything up, try selecting ALIGN from the EDIT menu. To move a field, select MOVE, then click twice on the field to select it.)

Fields
Contact.Contacts
Title.Contacts
Company.Clients
Street.Clients
City.Clients
State.Clients
ZIP_code.Clients
Job_no.Shoots
Shoot_date.Shoots
Fee.Shoots
Job_desc.Shoots
Detail_travel.Shoots
Exp_travel.Shoots
Detail_film.Shoots
Exp_film.Shoots
Detail_misc.Shoots
Exp_misc.Shoots
Fee_total.Shoots
Amount.Prints
Sales_tax.Prints
Print_total.Prints
Retouching.Prints

(continued)

You'll want to combine the "messenger" totals from both the "Shoots" and "Prints" files. Choose SET CALCULATION, call the calculation "msgr%", and enter "Messengers.Shoots + Messengers.Prints" as your calculation. (For non-programmers, the "%" sign says you want a numeric result.)

While you're in CALCULATION, create one for today's date. Call it "billdate$"—the $ sign shows you want a string, or text, result—and enter "date$(today)" as the calculation. Remember that when you use the form in SBPro, this calculation will take the current date format.

Next, enter the text. Using a different pen color will help differentiate the text and fields. The pen color can be changed with ATTRIBUTES. (You can always change it again later.)

INVOICE
Inv. No.:
Date:
P.O. No.:
Job No.:
Date:
Shooting:
Expenses:
Travel:
Taxis:
Meals:
Telephone calls:
Film & Processing
Misc. Assistant:
Prints: ["Amount" field]
Messengers: ["msgr%" calculation]
Retouching:
Subtotal: ["Print_Total" field]
Sales Tax:
TOTAL: ["Jobtotal%" calculation]

You need one more thing to make your invoice complete—the form to add up the total due. Call this calculation "Jobtotal%", and enter "Fee_total + Print_total" as your calculation.

Experiment with boxes, lines, colors, and text styles until you have a form you like, and which fits your letterhead or invoice.

Before you can use the form, you must establish links between the files so the form can find the information it needs. Select LINK from the SET menu. "Clients" will be your master file. Click on "Company," then "set." Make "Company.contacts" the secondary link, click on "set," and those files are linked. You'll also need these links:

Contact.shoots = Contact.contacts

Job_no.prints = Job_no.shoots

Now SAVE the form. (If you want to save a half-finished form before the links are established, you can. It just won't be much use yet.)

When you first load SBPro, you can simply open the form; it in turn opens all the files. This saves a lot of mouse-to-menu trips!

Be warned—if you edit the file, changing the name of any field used in the form, use the form editor to bring the form up to date before you load it into SBPro. On my copy an incorrect field name in a form brought on a mysterious message about an unsaved text file, closely followed by the guru. No doubt this bug will soon be squashed, but nevertheless...

You will probably want to design other forms of your own. You can use a single form to enter data into several files at once. A "job record," for example, could include details of the shooting, any prints ordered from it, and maybe even a digitized version of a typical picture or contact sheet.

One warning, based on frustrated experience: if you intend to include either an image (like a logo) or an external image field in your form, set the PROJECT RESOLUTION option to allow enough colors for your image (16 maximum) before you begin.

Once you enter *anything* on your form, you can no longer change the resolution.

I hope this will be changed in future versions of the program. (Precision Software, are you listening?)

PROGRAMMING SUPERBASE

Programming with DML (Database Management Language) can be as simple or as complex as you want it to be. Unlike most other application development programs, SBPro does not depend on a complicated language to get results; you can create a complex application using just the mouse and menus.

DML adds an extra dimension. Everything you can do with menus you can do through DML. And you can take it much further, creating your own requestors (called "dialogs" in the manual) and menus for a fully customized application.

To introduce you to DML, we'll set up some help files for certain specific fields, using the text editor. Then we'll have the "Help" key execute a program which asks us to select a field and then loads and displays the help file (if any) for that field.

From the PROJECT menu, select TEXT and create these help files. (You don't have to type in every word.) Don't forget to SAVE each file before starting a NEW one.

Save as "Date_enteredhelp"

Notice that Date_entered is a read-only constant field. It enters today's date automatically when you create a new record, and does not allow you to change it. You can, of course, edit the file (PROJECT EDIT FILE) so you will be able to make changes.

Save as "Telephonehelp"

The Telephone field is a multiple-response field. When you first enter a telephone number and hit RETURN, you will be presented with a blank field again. Don't worry—your number was entered. If your contact has an alternate phone number, enter it as the second response. Otherwise, press RETURN to leave the field. To see other responses in a record, press CTRL-N (for "next") or CTRL-P (for "previous").

Save as "Invoice_nohelp"

Enter your invoice number. When you become more familiar with SBPro, experiment with the LOOKUP function to check payments against original invoices.

Save as "Sample_photoshelp"

Sample_photos is an external image field. When you fill out a new record, type the full name of the image file, including any extensions, into the field. To see the image itself, click on the external file button (extreme right), and the image will appear on a separate screen. You can drag that screen down and resize the main screen to see both record and image at the same time. (Details in Manual 1.)

With the help files done, the next step is to create the program itself. From the PROGRAM menu, select NEW and type in this program. Be careful with capital letters—DML is case-specific in some areas, including program labels.

```
start:
CLS
a% = 0:a$ =""
REQUEST "Help for which field?","",5,a%,a$
IF a% = 0 THEN END
ON ERROR GOTO check
b$ = (a$ + "help")
LOAD TEXT b$:? TEXT: END

check:
If ERRNO = 68 THEN REQUEST "No help
available for this field ...", "Another
field?",1,a%,a$
IF a% = 0 THEN END
IF a% = 1 THEN RUN
END IF
? ERR$(ERRNO)
END
```

SAVE the program (from the PROGRAM menu, not the PROJECT menu) as "Photohelp". SBPro will add the extension ".sbp."

One more step—we have to set up the "Help" key to RUN the program. From the SET menu, select FUNCTION KEYS

(continued)

EDIT, click on "help" and type:

run "Photohelp" [RETURN]

All done. When you press "help," the program will load and run.

You might also want to add these options to your function key file as programming aids:

F1 OPEN FILE "Clients"
F2 OPEN FILE "Contacts"
F3 OPEN FILE "Shoots"
F4 OPEN FILE "Prints"
F5 OPEN FILE "Income"
F6 CLOSE "
F7 CLOSE ALL
F8 RUN
F9 EDIT
F10 MEMORY

Save the function key file on your data disk as "FunkeyPhoto." Now you can make life easier by setting up a "Start" program, like this:

DIRECTORY "df1:" [The directory for your data disk]
LOAD KEY "FunkeyPhoto"
OPEN FORM "Invoice"
DATEBASE "mm/dd/yy"
SET BUFFERS 20
SET RECORD ON
VIEW

Name it "Start" and save it in the same directory as your SBPro program. When you load SBPro, it will load this program, which will in turn set your current directory, load the function key file, open the "Invoice" form (which will open its associated files), set the date format and buffers, then switch to RECORD VIEW and display the first record—all while you finish your cup of coffee.

AIDS FOR SUPERBASE PROFESSIONAL

Faster screen text
To speed up text on the screen in Superbase Professional (and other programs I've worked with), try a public domain program called "Blitzfonts." It's on Fred Fish disk #60. The difference is amazing.

Faster disk access
FaccII from ASDG can make a real difference in floppy disk access speed if you find yourself calling up the same records over and over again. It's well worth the $35.

If you have enough memory to copy your files into RAM, preferably recoverable RAM, all the better. In fact, SBPro will do it for you, then copy them back, with your OK, when you close the file. Be careful to close your files and copy them to disk before using the QUERY, UPDATE or EDIT FILE functions— they have brought on the occasional guru for me, taking my RAM files down with them.

Fitting pre-printed stationery
When you get to designing forms, especially if they have to fit pre-printed material—invoices, etc.—buy a word-processing ruler. (You can get it in large stationery or art-supply stores.) It's made of clear plastic, with yellow-marked strips to help you keep your place. More importantly, it's marked off in 6, 10, and 12 lines to the inch, as well as the normal eighths of an inch. With it, you can measure your form quickly and easily.

Function keys
SBPro has a function key facility built in, but the Amiga version allows for only 21 keys—the ten function keys, the same keys shifted, and the Help key (F0).

However, there is at least one shareware function key program—"FuncKey"—which provides 50 keys (the 10 function keys in combination with the shift, ALT, LEFT-AMIGA and RIGHT-AMIGA).

Note the shareware program overrides the SB keys facility, with the exception of the help key. Also note that the Superbase key function enters commands directly; with the shareware program you must first open the command line. If you have a lot of repetitive text, you'd probably prefer the shareware program. If you want to use programming commands a lot, stay with SBPro's own keys facility.

Erasing in Data Entry
A small but useful undocumented extra allows one or two of the text editor commands to overlap into the data entry mode. For example, CTRL-E will erase from the cursor to the end of the field, and CTRL-W from the cursor to the end of the word (as opposed to CTRL-X, which erases the entire field). This is very useful if you use the RECORD DUPLICATE function to repeat similar records with minor changes. Use CTRL-Q or CTRL-U to undo the effects of these commands, returning the field to its original state.

Copying or Renaming Database Files
There are no commands for copying or renaming database files. (The COPY and RENAME options work on individual files, not on the set of files that make up a "database file." However, you can do it very simply with the REORGANIZE function. SBPro's REORGANIZE creates a new, clean file under a new name, with all deleted records and fields removed. If you want, you can then REMOVE the old database file, or keep it and have a new copy.

So there you are. A complete Superbase Professional application, one that you can—and should—adapt for your own use. Good luck!

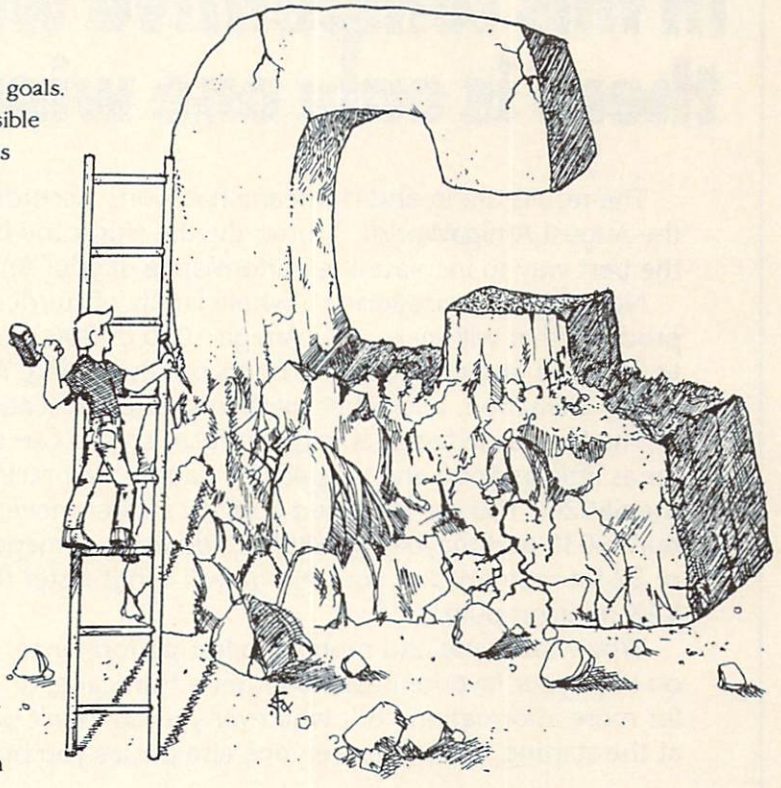
•AC•

On the Crafting of Programs

by David J. Hankins

This series of articles will explore various topics relating to the crafting of programs—programming style, structured programming, suggested program format, optimization, and so on. Since I do most of my programming in assembly and C, readers may notice a bias toward these two languages.

Without further ado, on to this month's topic...optimization.



Optimization concerns rewriting programs to achieve specific goals. Programs may be optimized by making them as small as possible or by decreasing their execution time. This month, I will focus on the latter goal—decreasing a program's execution time.

The first step in optimizing a program for speed is to determine what routines are responsible for overall performance. Without this knowledge, we could spend many hours modifying routines without improving the program. We need a method of determining which routines merit our attention. Fortunately, a tool exists which enables us to do just that. The tool is called a profiler.

A profiler measures the performance of routines within a program. Most profilers just count the number of times a routine is executed. Using this type of profiler, we would probably rewrite the routines that are executed most often. But what if these routines are already efficient? Just because a routine is executed many times doesn't mean that it is responsible for the program's overall speed. However, another type of profiler exists which actually measures the aggregate amount of time spent executing each routine in a program. Now we're getting somewhere!

If you program in Manx Aztec C on the Amiga, you're in luck. A free set of profiler utilities for Manx C programs is available through the public domain. These utilities, called p1 and p2, were developed by Tom Rokicki of Radical Eye Software. I got my copy of the profiler through CompuServe in data library 10, under the name profil.arc. My thanks to Tom for creating such a useful tool.

To profile a program using p1 and p2, we must first link our executable using the -t option with the Manx linker, version 3.30e or later. This causes the linker to produce a symbol file for our program. The symbol file has the same base name as the executable

(continued)

program with the extension ".sym" tacked onto the end. For example, if we linked using the -t option to produce the executable program "foo" (Why does everyone use the name "foo" for silly examples?), the linker would produce a symbol table for foo called "foo.sym." Next, we need to run p1 on the program we wish to profile. Continuing with our example, we would now type

```
p1 foo
```

P1 creates a modified executable program using the information contained in the symbol file. The modified program has the same name as the original program, except that the extension ".exe" is appended. Thus, the command "p1 foo" produces a modified executable file called "foo.exe."

Now for the fun part—profiling. To profile foo, we now need to enter the command

```
run p2 foo
```

This command sets up the profiler p2 and tells it to profile foo when it is next executed. To profile foo, simply execute the modified executable by entering

```
foo.exe
```

When foo.exe finishes execution, we click on the "finish" gadget in the window opened by p2. Doing so causes p2 to create a profile for our executable program. The profile has the same base name as the executable program, but ends with the extension ".mon." Thus, the profile for foo would be called "foo.mon."

All right. Let's profile an actual program.

Listing 1 contains the C source for the program "hello." When run, "hello" simply types "Hello world." to the screen. Because of the simplicity of the "hello" program, we probably wouldn't bother profiling it in the real world—how could we optimize such a small

program anyway without rewriting it entirely in assembly? However, by way of example, profiling "hello" illustrates how to use p1 and p2.

First, compile "hello" by entering

```
cc hello.c
```

and then link it by entering

```
ln hello.o -lc -t
```

We should end up with an executable, "hello," and a symbol file, "hello.sym." Part of the file "hello.sym" is shown in Listing 2. The symbol file provides offsets that indicate where a routine is located, and then lists the routine's name.

To profile hello, we first run p1 on it,

```
p1 hello
```

which produces the modified executable "hello.exe." Next, we enter the command

```
run p2 hello
```

to set up the profiling program. To create the profile for "hello," we type

```
hello.exe
```

and when the program is done, we click on the "finish" gadget in p2's window. Simple, huh?

The profile for "hello," "hello.mon," is shown in Listing 3. The profile lists each routine in the "hello" program, ranked in descending order, according to percent time spent in routine execution. Referring to Listing 3, the percent time is shown in column 3. Column 1 lists the number of times a routine was called; column 2 lists how many milliseconds it took (on average) to execute the routine each time it was called.

If a routine calls other routines, called routines are referred to as children, while the calling routine is known as the parent. Column 4 lists the number of milliseconds it took to execute the parent, plus the number of milliseconds it took to execute all child routines called by the parent. Column 5 shows the percent time spent in a parent, plus the percent time spent in its children. Finally, column 6 shows the name of the parent routine.

Back to our example—the "hello" program. The Manx C function printf( ) calls the function write( ). As shown in Listing 3, most of the "hello" program's time is spent in the write( ) function. This makes sense—after all, the "hello" program's purpose is to write "Hello world." to the screen.

At this point, you might wonder how p1 and p2 work. How do they manage to time each routine? In Manx C, functions (i.e. routines) begin with a link instruction (which is an MC68000 assembly language instruction) and end with unlink (another MC68000 assembly language instruction). To show this, I ran the DSM disassembler on the "hello" program. A partial listing of the disassembly is shown in Listing 4. The "hello" program begins in Listing 4 at the label "_main." From the listing, we see that the first instruction in the main() function is link, and the last instruction before the function returns is unlink.

When p1 is run, it modifies the target executable program by replacing the link and unlink instructions in each function with a trap #3 instruction. The trap #3 instruction causes the Amiga's MC68000 CPU to jump to the third trap vector. (In MC68000 terms

(continued)

this is known as an exception. On other processors, this type of occurrence is referred to as an interrupt.) Thus, when p2 is run, an exception occurs each time a function is entered. At this point, P2 gains control of the target program and starts a timer for the function. After the timer starts, p2 returns control to the target program. Upon exiting, the function another exception occurs. Once again, p2 gains control of the program and notes the elapsed time spent in the function. In this fashion, p2 is able to keep a running total of the time spent executing each function. Pretty neat stuff!

In next month's article, I will present a real-world example of how to use the profiler. We will profile the DSM disassembler, version 1.0c, to see which of its routines can be easily changed to increase overall disassembly speed. Then, we will modify one or two routines based on the information provided by the profile. Finally, we will measure the increase in disassembly speed attributable to our modifications. So, until next month:

```
#include "stdio.h" main(){ printf( "Goodbye world!" ) ; }
```

About the author:

Mr. Hankins is an avid Amigaphile, having purchased an Amiga 1000 when they first came out in 1985. Recently, Mr. Hankins formed a company called OTG Software which produces DSM, an MC68000 disassembler for the Amiga. Readers wishing to contact Mr. Hankins can reach him on BIX [dhankins] or Compuserve [76515,1650].

Listing One
C Source (Hello.c)

```
/*********************************************************
 *                                                       *
 *      Program:  hello.c                                *
 *                                                       *
 *********************************************************/

#include "stdio.h"                  /* I/O header file    */
#define  NORMAL   0                 /* Normal execution   */

void
main()
{
   int        printf() ;            /* C function printf  */

   void       exit() ;              /* C function exit    */

   printf( "Hello world.\n" ) ;
   exit( NORMAL ) ;
```

Listing Two
Symbol File for Hello

```
Segment 00:  Hunk 000
          00000000 __H0_org
          00000002 _main
          0000002a .begin
          0000009c _geta4
          000000a4 __main
          000001dc __cli_parse
          000003a4 _strcat
          000003aa _strncat
          000003ce _strncpy
          000003f0 __wb_parse
          000004a4 _printf
          0000054e _format
          0000086a .divs
          00000892 .mods
          000008ac .modu
          000008b8 .divu
          00000912 _putchar
          00000928 _aputc
          0000096a _putc
          000009ce _fclose
          00000a52 _flsh_
          00000b34 _newstream
          00000b6c _getbuff
          00000bfc _lmalloc
          00000c3c _malloc
          00000c50 _free
          00000c9c _isatty
          00000cf4 _unlink
          00000d18 _write
          00000d96 _Chk_Abort
          00000dc6 __abort
          00000df0 _exit

        Remainder of symbol file omitted.
```

Listing Three
Profile of Hello.

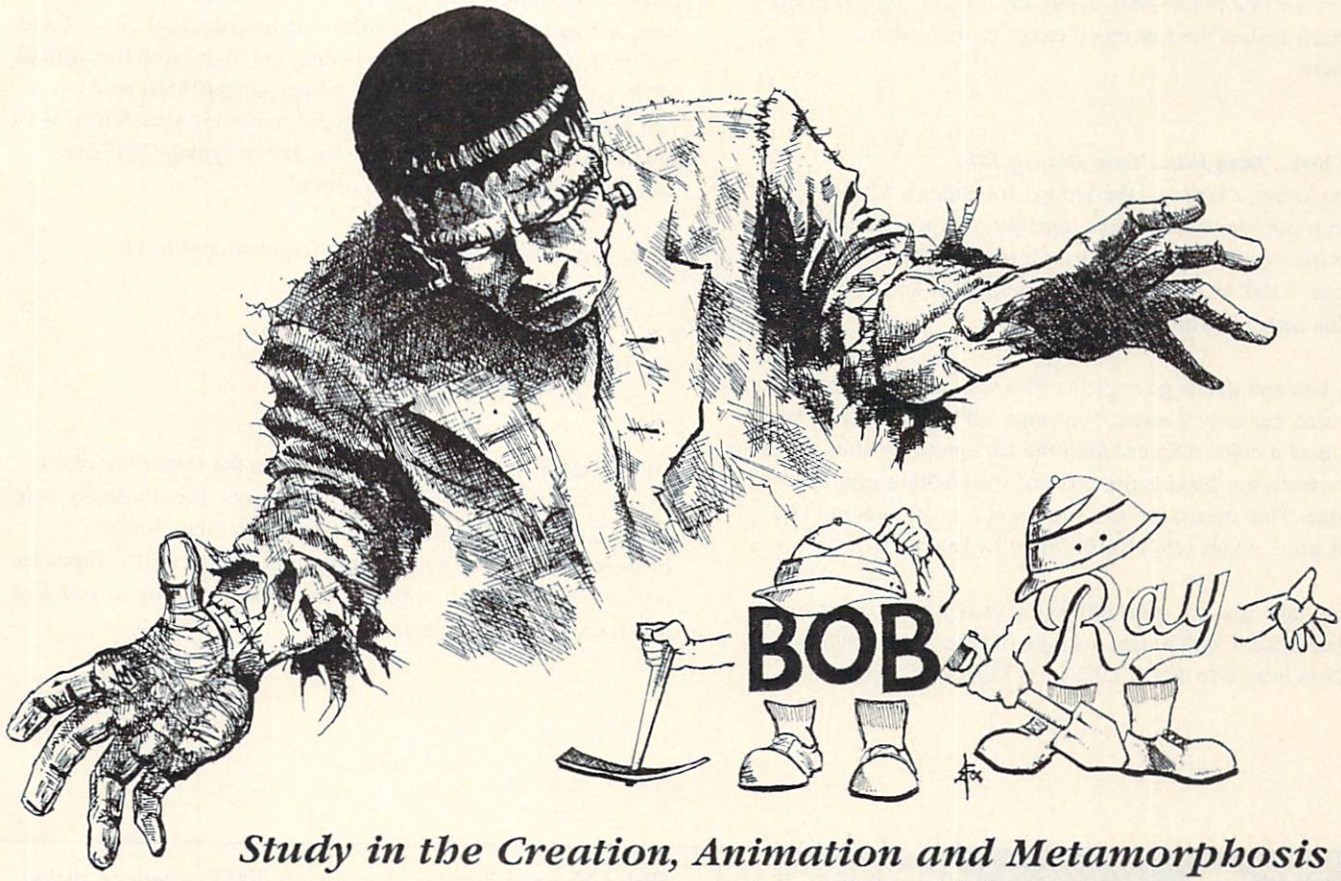
```
                Self          Self + Children
# Calls   ms/call   %time   ms/call    %time   name
------  ---------  -----  ---------  -------  ----
     1     87.52  76.44      87.72   76.61   write
    23      0.23   4.67       0.23    4.67   close
     1      3.84   3.36     111.90   97.72   _main
    20      0.15   2.62       0.32    5.52   fclose
    13      0.21   2.39       7.24   82.19   aputc
    13      0.21   2.33       7.44   84.52   putchar
     1      2.09   1.82       8.41    7.34   exit
    13      0.13   1.48       0.26    2.96   putc
     1      1.56   1.37      98.34   85.89   format
     1      0.91   0.80       0.91    0.80   _cli_parse
     1      0.72   0.63       0.72    0.63   lmalloc
     1      0.59   0.51       0.59    0.51   free
     2      0.26   0.46      44.84   78.32   flsh_
     1      0.33   0.28       1.43    1.25   getbuff
     1      0.26   0.23       0.26    0.23   isatty
     1      0.20   0.17     107.14   93.57   main
     1      0.20   0.17       0.20    0.17   Chk_Abort
     1      0.20   0.17      98.54   86.06   printf
     1      0.13   0.11       0.85    0.74   malloc
     0      0.00   0.00       0.00    0.00   unlink
     0      0.00   0.00       0.00    0.00   newstream
     0      0.00   0.00       0.00    0.00   _abort
     0      0.00   0.00       0.00    0.00   _wb_parse
     0      0.00   0.00       0.00    0.00   _exit
```

Listing 4
Partial Disassembly of Hello.

```
*****************************************************************
*     DSM MC68000 Disassembler Version 1.0d (06/01/88).         *
*     Copyright (C) 1987, 1988 by OTG Software.                 *
*     All rights reserved.                                      *
*     Disassembly of :  hello                                   *
*****************************************************************
          SECTION  segment0,CODE
seg0
__H0_org
          bra.s    .begin

_main     link     a5,#$0000
          pea      L3(pc)
          jsr      _printf(pc)
          addq.w   #$4,a7
          clr.w    -(a7)
          jsr      _exit(pc)
          addq.w   #$2,a7
          unlk     a5
          rts

L3        dc.b     'Hell','o wo','rld.'
          dc.b     $0a,$00

.begin    bsr.s    _geta4
          lea      -$7da6(a4),a1
          lea      -$7da6(a4),a2
          cmpa.l   a1,a2

Remainder of startup code for Manx C omitted from listing.
```

•AC•

BOB & RAY MEET FRANKENSTEIN



Study in the Creation, Animation and Metamorphosis of Graphic Objects in AmigaBASIC

by Robert D'Asto

One of AmigaBASIC's most enjoyable aspects is its ability to create and manipulate graphic blocks and animated objects with the GET, PUT and OBJECT commands. These commands are easy to learn and use just as they are, with no further understanding of them other than their respective syntax and parameter requirements. We simply dimension an array, "GET" a graphic into the array and "PUT" it anywhere on the screen we wish; or we create a bob with the Object Editor, "feed" it into the OBJECT.SHAPE command, and off we go with the OBJECT animation commands.

It's fun, no doubt about it, but there's a rub: the graphic block must be drawn on the screen before we can GET it and use it in the program, and the bobs must be created separately and stored on disk before they can be accessed and used in our little binary dramas. How can we share our masterfully rendered and deftly animated thingamabobs with the world without all that onscreen initialization of graphics at the beginning of our programs and those annoying pauses for drive machinations and "File Not Found" error messages while loading bobs from disk?

Isn't there a faster, more direct way of creating graphic objects with AmigaBASIC?

Lament no more. Did you know that graphic arrays can be created directly from data within your own source code without drawing them on the screen? And that bobs can also be created directly within the program, without using the Object Editor at all? Did you also know that once created, graphic array blocks can be "metamorphosed" into bobs and then animated with all the OBJECT commands? Or that once you create them, you can alter the shape, color, size, and orientation of all objects to suit your fancy?

Yes, you can do all these tricks, and more, with AmigaBASIC and a quick and painless review of the "innards" of the Amiga and a few commands. In case you're wondering, this has nothing to do with LIBRARY commands or ROM Kernel routines and does not require handling massive programming volumes or other blunt instruments. All you need is a good grasp of a few basics and a little practice. The rest is up to your own imagination and cleverness.

We'll start with something you may already know: the manner in which graphic arrays are created. If this part is new to you then fire up your AmigaBASIC editor and tap along with the little code fragments given here so you can see the results and quickly join the ranks of the enlightened few.

(continued)

The "secret" of mastering our graphic creations lies in understanding the basic "building blocks" the Amiga uses to create them in the first place—bits. As in any subject, a good grasp of the basics makes the rest much easier to understand, so let's start here.

'Dem Bits...'Dem Bits...'Dem Binary Bits

As you know, a bit is a binary digit. It's either a "1" or a "0". You can put two or more bits together to represent any number you wish. For example, the binary number "1100" equals 12 because "1100" in binary means "one 8, plus one 4, plus zero 2s, plus zero 1s" which equals 12.

Now, bits and pixels go together like fleas and Fidos because a pixel also has only 2 states, "on" and "off". "On" means the pixel is of a color different than the background—thus is visible. "Off" means the pixel is the color of the background—so it is invisible. That means we can represent a pixel pattern of "2 pixels on, 2 pixels off (1100)" simply by saying "12".

We can take this one step further by using the hexadecimal number system which uses a single digit to represent all numbers from 0 to 15 : 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

Now the binary number "1100" (12) can be represented by the single digit "C", and a 16 bit pattern of 1100110011001100 (52,428 in decimal) can simply be stated "CCCC" in hex. How does AmigaBASIC know we mean the hexadecimal "CCCC" and not the letters "CCCC"? By preceding the digits with the symbol "&H", so it becomes "&HCCCC" which AmigaBASIC reads as a hex number. There's even a little programming trick which we'll get to shortly that saves you the trouble of typing "&H" over and over when using hex statements.

In the meantime, type in this little fragment and run it:

```
SCREEN 1,320,200,2,1
WINDOW 2,,,0,1
LINE (0,0)c(15,9),3,bf
```

This produces a little orange rectangle in the upper left of the screen. Notice the dimensions of this object. It is 16 pixels wide (0 to 15 = 16) and 10 pixels high. It is also 2 bits "deep" because the SCREEN statement set up a screen with 2 bitplanes. If the subject of depth or bitplanes seems confusing to you, fear not. It's simpler than it sounds.

Depth can be thought of as "color dimension", and bitplanes as the unit of measurement of that dimension. A depth of 1 bitplane gives 2 possible colors because there are 2 possible conditions of 1 bit (0 or 1), a depth of 2 gives 4 possible colors because there are 4 possible combinations of 2 bits (00,01,10 or 11) and so on. More on this in a moment, too. For now, add this routine to the above listing and run it:

```
DIM array%(22)
GET (0,0)c(15,9),array%
PUT (100,100),array%
```

Okay, now we've "captured" our little object in a graphic array and PUT it somewhere else. Nothing new so far...but wait! Let's take a peek inside that array and see what's lurking there. Add this to the routine and run it:

```
HexDump:
FOR x=0 TO 22
element$=HEX$(array%(x))
WIDTH 30:PRINT element$ + ",";
NEXT x
```

This little loop converted the array's 23 elements (0 to 22 = 23) into hex values and printed them on the screen, separated by commas. This is called a "hex dump" of the array. It's how the computer "thinks" of our object.

The Amiga builds up its graphic images with "bricks", each made up of 16 horizontal pixels stacked together like a brick layer makes a wall. Our object consists of 10 of these "bricks", stacked up to form a rectangle. If the object were a triangle, each block would still be 16 pixels wide, we would just "chip away" parts of the "bricks" by turning some of the pixels into the color of the background so they would be invisible.

Take a look at the printout of our graphic elements. What do these things mean? The first element (element 0) is "10", which means "16" in hexadecimalese (one "16" plus zero). That tells our GET and PUT commands the object is 16 pixels wide, because the first element in a graphic array is always its width. The next element is the height of the object is a hex "A" which means 10 or 10 pixels high. The third element is the "depth" which is "2".

The format of graphic arrays always uses the first 3 elements (elements 0 through 2) to state the width, height and depth of the graphic in that order. All the following elements represented the pixel arrangements in the "bricks" that make up the overall shape of the graphic. Each element following element 2 represents a single, 16 pixel "brick".

If our object is made up of only 10 "bricks" why are there 20 shape elements in the array? This is where the bitplanes come in. If only 10 elements described the object the Amiga would

(continued)

know its shape but still wouldn't know what color to make the individual pixels.

It "pretends" the object is made up of "layers" in addition to "bricks". That is, it has depth. The first 10 shape elements following the depth element give the shape of the 10 "bricks" that make up the "top" bitplane and the last 10 elements represent the "bottom" bitplane. If we were working with a 3 bitplane screen an additional 10 elements would represent that third "layer".

In our hex dump all the shape elements are the same. Each is "FFFF". In the hex system, "F" equals a decimal 15, which is 1111 in binary, so "FFFF" simply means "a row of 16 pixels". If we wanted our graphic to be a triangle we could work it out like this:

```
0000000100000000 = &H0100
0000001110000000 = &H0380
0000011111000000 = &H07C0
0000111111100000 = &H0FE0
etc...
```

Just take the pixels in each row four at a time and figure the binary value for them. This value will be somewhere between 0 and 15. Convert that value to the corresponding hex number (0 to F), and go on to the next four, etc. Each "brick" of 16 pixels can be converted to a 4 digit hex number in this way. When you get to the bottom, start again at the top of the bitplane "underneath" the first (more on this in a moment). Use graph paper to help keep track of the patterns if it gets too complex.

If you are dealing with a graphic wider than 16 pixels, simply take the upper left "brick" first, then the one to its right, and so on, to the right edge of the whole block. Then take the leftmost "brick" on the second row, etc. Continue down to the bottom right "brick", then start again with the upper left of the next deepest bitplane.

We also have a second "layer" of solid "on" pixels In our graphic array. The second layer represents the array's last 10 elements. Imagine the last 10 elements as sitting beneath the first 10. We can now think of each pixel as having a second pixel beneath it. This way we can control the color of each pixel with another binary "trick". If we could "see through" each pixel on the screen down to "the one beneath it" we would get another binary number comprised of the bottom and top bits. We're simply viewing the "pretended third dimension" of the graphic to get information about its color.

In our example, each of the "bottom" pixels are "on", as are each of the "top" ones. That produces the binary number "11" which equals 3 in decimal. Lo and behold, our rectangle is orange —the color "3" in the default PALETTE we're now using. The binary number derived from this "see through" trick will correspond to the PALETTE ID numbers, the "deepest" bitplane being the first digit of this binary number.

Suppose we wanted our object black instead of orange. Black is PALETTE 2, so we just change the bitplane arrangement so we have a binary 2 when we do our "see through the pixel" trick. Binary 2 is "10", so if we keep all the "bottom pixels on" and turn all the "top ones off" we will have a binary 2 (10) in our depth dimension for each pixel. It's as if we viewed the bitplanes "edge wise", with the bottom bitplane leftmost. We should end up with another rectangle of the same shape, but black instead of orange.

How do we change the values in our graphic array to achieve this? A piece of cake. We just enter a simple loop which runs through only the elements we want to change, and assign them a value of zero. Add the following code and run it:

```
FadeToBlack:
FOR x=3 TO 12
array%(x)=0
NEXT x
PUT (120,100),array%
```

Elements 3 through 12 make up the "top" bitplane of our object. This loop converts all these to 0, leaving the "bottom" bitplane as it was. When viewed "edge wise" we get the binary number

"10" (digital 2) for each pixel position, making each pixel the color of PALETTE 2, which is black in our default palette. We now have a second rectangle just like the first, except this one is black.

Now, here's a tricky application for just this sort of operation. Erase that last "PUT" statement from the editor and use this one instead:

```
PUT (104,104),array%,OR
```

You should now have what's called a "drop shadow" effect. It gives the object the illusion of 3D by causing it to "cast a shadow". It's not all that dramatic with our little object, but it can be a real knockout with more complex shapes and is actually the method used by commercial programmers to create 3D style fonts.

Let's review how we did that. The FadeToBlack loop changed all the values on the lower bitplane to 0, which gave us an identically shaped black rectangle. The new PUT statement positioned this black shape slightly below and to the right of our original shape, and the OR option made visible only that part of the black shape which was not directly coincident with the original shape. When viewed on the screen, this produced a convincing shadow effect.

You could also turn the object white by leaving elements 3 through 12 as they were and changing elements 13 through 22 to "0", you could create a checkerboard effect by alternating the FFFFs with zeros, or create any shape you like by simply changing these hex values to represent the desired bit patterns. Experiment. Just be sure you leave the first three elements (0 to 2) alone when you do this or you'll invite the Guru. You can change these first elements, of course, but if you do you'll also have to re-dimension the array to accommodate it and make the necessary changes in the shape data elements.

One more thing you need to know about graphic arrays is the method used to dimension them. I saved this for last because it's easier to grasp once you understand everything else. We know each element in the array contains the data describing 16 of its pixels...So do we derive the number of elements needed in the array by counting the pixels in the object and dividing by 16? That's close, but not quite it. Remember, we have to allow for the dimension elements and all the bitplanes, too.

Here's the easy way. First determine the width of the object in pixels and divide this number by 16. For example, if the width of your object is 100 pixels, dividing by 16 gives us 6 with a remainder of 4. Just drop the remainder and round the number up to 7. This means the object can be created within a width of 7 of our 16 pixel "bricks". Multiply 7 by the height of the object. If it's 100 pixels high the result would be 700. Multiply again by the depth. We'll say the depth is 2, so the result is 1400.

(continued)

Now all we need are the 3 elements which give the dimensions of our object, so we add 3 and get a grand total of 1403 elements. Since arrays always start counting at 0 instead of 1, we can dimension our array: DIM array%(1402). The reason for the % symbol at the end of the array name is to identify it as a short integer array. This isn't absolutely necessary, but it makes things much easier when doing hex dumps or manipulating elements.

By this time you may have gotten the idea that you didn't really have to draw the rectangle before you used the GET statement which "loaded" it into the array. You're right. In fact, you didn't have to use the GET statement at all. You could have written out and "stuffed" all those element values directly into the array and used PUT to position it where you wanted, bypassing the onscreen initialization of the object. Erase the program we just used (or save it for comparison) and try this one:

```
SCREEN 1,320,200,2,1
WINDOW 2,,,0,1
DIM array%(22)
FOR x=0 to 22
READ a$
element=VAL("&H"+a$)
array%(x)=element
NEXT x
DATA
10,A,2,FFFF,FFFF,FFFF,FFFF,FFFF,FFFF,FFFF,FFFF,FFFF,FFFF
DATA FFFF,FFFF,FFFF,FFFF,FFFF,FFFF,FFFF,FFFF,FFFF,FFFF
PUT (100,100),array%
```

Voila! The little orange rectangle should now have materialized in full view of all, created solely by source code.

Notice there was no onscreen initialization of the object. It just appeared, fully drawn, where you told it to. The "VAL("&H"+a$)" statement saved us a little typing. If we didn't use it, we'd have to precede each of the DATA values with the prefix "&H" so AmigaBASIC would recognize them as hex numbers. Instead, we simply "tack it on" within the loop and use the VAL function to provide the numerical values of each.

You should now have a good grasp of the basics of graphic array construction. I suggest playing around with the above principles for awhile, creating bigger and better objects and getting the hang of altering them by experimenting with different FOR/NEXT loops that change the element values. You can draw many different types of objects using the "stock" AmigaBASIC graphic commands, public domain and commercial paint programs, or even draw one pixel at a time with the PSET command. Then you can use a "HexDump" routine like the one listed above to get the hex values and take it from there.

Assuming you're not thoroughly confused at this point, we'll now move on to the more exotic possibilities of altered states in AmigaBASIC—bobs.

Normally, when we want an animated object in our programs we boot up our trusty Object Editor, render it, save it to disk and then "invoke" it with an "OPEN FOR INPUT AS" statement from within our program, which resurrects it from disk where it can then break the surly bonds and soar aloft into another silicon fracas. Well, I hate to be the one to tell you, but...tough programmers don't need the Object Editor!

Well...not always. You see, you can carve your own bobs out of the same stuff we used to make the above graphic array, and in much the same way. All you need to know is the format that AmigaBASIC uses to construct these objects. Then you can glue them together from a READ/DATA loop very similar to the one we used for our simple graphic rectangle earlier.

Just in case you don't believe me, clear off your editor and try out this little baby:

```
SCREEN 1,320,200,3,1
WINDOW 2,,,0,1
FOR x=1 to 46
READ a
Obj$=Obj$+CHR$(a)
NEXT x
OBJECT.SHAPE 1, Obj$
OBJECT.X 1, 10
OBJECT.Y 1, 100
OBJECT.ON 1
OBJECT.VX 1, 50
OBJECT.START 1
Loop:GOTO Loop
DATA  0, 0, 0, 0, 0, 0, 0, 0
DATA  0, 0, 0, 2, 0, 0, 0, 16
DATA  0, 0, 0, 5, 0, 24, 0, 3
DATA  0, 0,255,255,255,255,255,255
DATA 255,255,255,255,255,255,255,255
DATA 255,255,255,255,255,255,255,255
```

You should now have a little orange box, which bears a striking resemblance to a bob, sailing eastward across your screen. It is a bob, and it's just as bob-like as if you'd made it with the Object Editor, and it will respond to all the OBJECT commands.

How did we do that? Again, it's simple, but it takes a little explanation. You'll notice no hex numbers are involved. This time the values are good ol' decimal numbers through and through. This is because AmigaBASIC doesn't use 16 bit "bricks" to construct bobs. It uses 8 bit ones, instead. Most of the values you see in the DATA statements above represent 8 bits or 8 pixels of the whole object. Since the maximum binary number attainable with 8 bits is 11111111, or 255 in decimal, the values that make up any bob will range from 0 to 255, with 255 representing a solid horizontal line 8 pixels long.

Our bob is 16 pixels wide (2 8 bit "bricks"), 5 pixels high and 2 bitplanes deep. If you look again at the DATA statements above you'll see these three values (16, 5 and 2), but in rather odd positions. The "2" (depth) is in the 12th position, the "16"

(width) is in the 16th position, and the "5" (height) is in the 20th position. That's because that is the format for indicating these dimensions for a bob. The OBJECT.SHAPE function expects to find these parameter values in the positions shown.

You've probably figured out by now what all the 255's are: 8 bit representations of the solid lines that make up the object. But, what are all those zeros, and what are the "24" and "3" doing there?

Let's start with the zeros. The first 8 zeros are always there. The OBJECT.SHAPE function expects to find them there. If it doesn't, it knows something is amiss and an "Illegal Function Call" error will be generated. The first 3 zeros on the second DATA line are there because the first four values of this line are actually reserved for the depth parameter. Since we only need the last one (the 4th) to represent the number "2", we let the first 3 remain zero.

The same is true for the width parameter that follows. The width of our object is 16 pixels, so we only need to use the last element on the line to assign that value. Moving to the third DATA line, we again use only the fourth element to assign the height of the object, so the first 3 remain zero and the fourth gives the height parameter.

Now it gets a little tricky. The "24" and the "3" and the first 2 zeros of the fourth line are decimal equivalents of 8 bit numbers AmigaBASIC uses to denote certain attributes of the object such as "collision plane" and "image shadow" and other stuff that we, frankly, don't need to know for our purposes here. This material is covered in the ROM Kernel Manuals, which is one of the things we wanted to avoid for now. Suffice to say, if you put the values "24" and "3" where you see them in the above DATA statements, and the two zeros at the beginning of the fourth DATA line, you'll have yourself a real, live bob, which behaves as a good bob should.

One other point in the routine that needs explanation is the "Obj$=Obj$+CHR$(a)" statement. When you make a bob with the Object Editor and save it to disk, each of the values that define it are saved as "Tokens", or single characters from the Amiga ASCII character set. The OBJECT.SHAPE command, when used "normally", expects to be given a string of these Tokens, which it then converts into the necessary binary data which defines the bob.

You may have noticed that no array was used in the above "bob maker" routine because none was needed. All the OBJECT.SHAPE command needs is a string of characters corresponding to the proper ASCII codes. So the above statement simply creates this string with the "+" operator.

I suggest you always arrange your DATA statements as I have done above, with 8 values on each line and parallel spacing. It makes it much easier to keep track of what you're doing. A larger bob would simply have more DATA lines containing the shape values for the additional 8 bit "bricks", arranged in the same order as is used with graphic arrays: left to right, top to bottom.

Congratulations. You now possess the secrets of graphic and object creation and manipulation. The time has come to try out our very own metamorphosis machine: a devilish device which can change an ordinary and lifeless graphic block into a living, animate bobcreature! Hideous, but true.

Clear the editor, lock the doors and put on your darkest sunglasses. Then, drop this little coin in the slot:

```
SCREEN 1,320,200,2,1
WINDOW 2,,,0,1
DIM graphic.array%(34)
DIM bob.array%(89)
CIRCLE (8,8),5,2
PAINT (8,8),3,2
GET (0,0)c(15,15),graphic.array%
bob.array%(11)=graphic.array%(2)
bob.array%(15)=graphic.array%(0)
bob.array%(19)=graphic.array%(1)
bob.array%(21)=24
bob.array%(23)=3
y=26
FOR x=3 TO 34
a=graphic.array(x)
R = a AND &HFF
L = (a AND &HFF00)/256
bob.array%(y) = L
bob.array%(y+1) = R
y = y+2
NEXT x
FOR x = 0 TO 89
bob$=bob$+CHR$(bob.array%(x))
NEXT x
OBJECT.SHAPE 1,bob$
OBJECT.X 1,50
OBJECT.Y 1,50
OBJECT.ON 1
OBJECT.VX 1,50
OBJECT.START 1
Loop:GOTO Loop
```

Barring typos, you should now be watching your little changeling chugging merrily across the screen. What had started out as a plain vanilla, screen doodle is now a genuine, animated, yessirree bob...awaiting your every command.

Take another look at the code. We made a simple image, "captured" it in a graphic array, and transferred the dimension parameters to the proper positions for a bob format in a second array (needed only as a temporary buffer).

Then we did something really tricky: we took each hex shape value from the original object and "lopped it in half" bit wise with two AND statements and put both halves in the appropriate positions in the second array. This trick takes care of the 16 to 8 bit conversion necessary between the two formats. It's also another method of manipulating bit patterns in general, and for creating other interesting effects within the objects themselves and... Well, as they say, that's another story.

In the meantime, I suggest you experiment with these principles. Practice creating different graphic patterns and try loops that change the element values, and observe the results. Play with bobs, too. Make 'em, un-make 'em, shake 'em, and bake 'em. Don't be timid...discombobulate those bits!

You'll learn a lot about programming and your computer in the process, and that's what it's all about!

•AC•

1> **by Rich Falconburg**

The Command Line

The Continuing Guide to the CLI

We have now explored two methods of creating a new console (CLI) window. For keyboard fanatics, the Workbench route is a pain. Leaving the original (AmigaDOS) CLI window open works, but might not be preferred. The NEWCLI command provides the third and most flexible method for creating a new console window. In its simplest form, NEWCLI creates a new window using some default parameters.

```
1> NEWCLI
```

This causes a console window to appear with the prompt indicating the number of the CLI process created. This new window may be customized by providing the NEWCLI command with some parameters. The definition template is:

```
NEWCLI CON:x/y/Width/Height/Title (FROM)
```

where:

CON: = defines a "console" CLI window.

x/y = defines the top left pixel coordinates.

Width/Height = window size in pixels.

Title = name of the window

FROM = is an optional script file to be executed when the window is created. A CLI "startup-sequence".

NOTE: Valid pixel values are 0 to 640 for X and Width, 0 to 200 for Y and Height on standard Workbench (0 to 400 on an Interlace Workbench). If you use spaces in the Title field, you must enclose the entire specification in quotation marks. The FROM parameter can be used any of the following ways:

```
NEWCLI CON:0/0/640/200/My_CLI FROM CLI.config
NEWCLI CON:0/100/200/100/Console S:CLI.startup
NEWCLI FROM SYS:CLI.stu
```

The script file must be EXECUTE compatible, which allows you to customize each CLI process. The PROMPT command might be useful here. With it, you can change the benign AmigaDOS prompt to something with more flair. The string between quotes becomes the new prompt for that CLI window. For example, I use the following line for a CLI window that I always leave open. I consider it my system Console. It can be something simple like:

```
PROMPT "CONSOLE-> "
```

To display the CLI process number, use the special character combination %N in the string.

```
PROMPT "(%N) CONSOLE> "
```

Or you can go all out:

```
PROMPT "*e(7;32;43mCONSOLE*e(0;31m> "
```

Whoa! What's all this? First let's discuss ANSI Control Sequences. Many alpha-numeric terminals recognize a set of character combinations that do specific things. If you are familiar with terminal emulators, you have probably noticed reference to VT100 or VT200 emulation. Others exist, but these are the most common.

The VT100 or VT200 terminals are manufactured by Digital Equipment Corporation for use with their various computer systems. These terminals recognize specific combinations of characters for performing special functions, including clearing the screen, homing the cursor, and displaying various character attributes (reverse, blink, underline, bold).

Other character combinations produce double height/double width characters and character graphics. The Amiga's console window provides similar support. These special sequences use "escape sequences." ESCape is designated by "*e[" in the PROMPT string above. The values that follow the escape character change the foreground and background colors for the word "CONSOLE" and print it in reversed video. The second sequence resets everything. The escape sequence above could be described as:

(continued)

ESC S;F;Bm (ESC = *e( )

where: S = Style
0 = PLAIN, 1 = BOLD,
3 = ITALIC, 7 = INVERSE
F = Foreground color 30 - 37
B = Background color 40 - 47

The numbers for the colors equate to various combinations of the Workbench colors as set in Preferences. Other useful sequences are:

Cursor Control

ESCnB	Cursor Down
ESCnA	Cursor Up
ESCnD	Cursor Left
ESCnC	Cursor Right
ESCr;cH	Move Cursor to Row r, Column c.
ESCH	Home the Cursor (top left corner)
ESC0 p	Turn Cursor off (space intentional)
ESC p	Turn Cursor on (space intentional)
ESCnP	Delete characters to the right
ESCJ	Erase to the end of the display
ESCK	Erase to the end of the line
ESCM	Delete line

Window Control

ESCnu	Set line length (Not ANSI)
ESCnx	Set left margin (Not ANSI)
ESCnt	Set page length (Not ANSI)
ESCny	Set top margin (Not ANSI)
ESCnT	Scroll lines down
ESCnS	Scroll lines up

For each sequence above, the trailing letters are case sensitive, and "n" is an integer value and is optional. If left out of the window control sequences, "n" is reset to the default value. Each of these escape sequences can be used with the ECHO command if the string is enclosed in quotes.

Close That Window!

To remove any CLI console process, use the command ENDCLI.

```
4> ENDCLI
CLI task 4 ending
```

Some programs may keep the console window open. However, the window no longer accepts input, and when the program keeping the window open terminates, the window closes. To determine if the process has terminated, use the STATUS command. To determine which CLI processes are currently executing in the system, type:

```
2> STATUS
Process  1: No command loaded
Process  2: Loaded as command: STATUS
Process  3: Loaded as command: RUN
```

To display information about a single process, enter its number following the command.

```
2> STATUS 3
Process  3: Loaded as command: RUN
```

More detail can be displayed with the TCB (Task Control Block) or FULL parameter.

```
2> STATUS FULL
Process  1: stk 1600, gv 150, pri 0 No command loaded
Process  2: stk 1600, gv 150, pri 0 Loaded as command STATUS
Process  3: stk 3200, gv 150, pri 0 Loaded as command RUN
```

or

```
2> STATUS TCB
Process  1: stk 1600, gv 150, pri 0
Process  2: stk 1600, gv 150, pri 0
Process  3: stk 3200, gv 150, pri 0
```

Broken down this is:

stk	= Stack size
gv	= Global Vector size
pri	= Priority

The Global Vector and Stack size are part of the handler process for the CLI and are used by the system. Do not confuse this stack size with the one shown and created by the STACK command. The value in the information given by the STATUS command is the smallest stack size permissible in each process. To see the current value, enter STACK without parameters. The returned value is memory set aside for temporary storage of variables and other information used by the various commands. Some programs require larger stacks than the default value. To change stack size, enter STACK and the number of bytes for the size.

```
2> STACK 10000
```

The system uses the Priority value to determine how big a slice of the CPU's time each process is allowed. Values range from -128 to +127 and may be changed with the CHANGETASKPRI command. Positive values increase the priority of the process (more CPU time); negative values decrease the priority (less CPU time).

Although multitasking makes it look like everything is happening at once, in reality, the operating system steps through the process list, running each process in turn for a specified period of time. It then suspends execution of that process and stores pertinent information to a holding area where the system can find it for the next slice. This greatly oversimplifies the actual chain of events, but is close enough to explain what the Priority does. The length of time allowed for execution is partly determined by the priority.

CHANGETASKPRI is good to use with background tasks. Version 1.2 of Workbench only allows changing priority of the current task. That is, the CLI from which the CHANGETASKPRI command is issued is the process that has its priority changed. Increments of no more than -5 or +5 should be used. Setting the priority of a busy process too high locks you out of the system until it completes. Setting one too low can degrade system performance as well. To set the priority of a batch process to -5, insert the following line near the beginning of the script file.

```
CHANGETASKPRI -5
```

Now Stop That!

Occasionally a program running in the background must be aborted. If there is a way to do so within the program, use it. Sometimes this is not possible or the program ignores the attempt. If the program was started from the CLI, you may be able to force termination. Some programs recognize Control character sequences during operation. (The most common of these is Control C, the ASCII Break character.) AmigaDOS recognizes Control D as a batch file abort command. You send these characters to the offending process with the BREAK

command. First, use the STATUS command to determine the CLI process number. Let's use CLI number 5 for our example. To abort, enter:

```
1> BREAK 5
```

This operation sends the Control C sequence to CLI process number 5. This is the same as clicking in the console window for CLI 5 and, while holding down the Control key, pressing the C key. Other sequences BREAK can send are Control D, Control E, and Control F. If you're not sure which characters might be recognized, use:

```
1> BREAK 5 ALL
```

The ALL parameter causes all control characters mentioned to be sent to task 5. This operation may not work on all programs, but you won't hurt anything by trying.

Operator Error?

If you use the CLI often, you'll eventually get some vague or cryptic messages from AmigaDOS complaining that it can't do

(continued)

something. Sometimes you can get a little more information by entering the WHY command immediately after the failing command. Usually, you get this:

```
3> WHY
The last command did not set a return code
```

On the other hand, you might get some helpful information.

```
3> WHY
Last command failed because object in use
```

For reasons we may never know, failed commands provide really useful numbers, instead of just printing the text associated with the error. Fortunately, we have the FAULT command to translate this garbage into something we can understand.

```
2> FAULT 212
Fault 212: object not of required type
```

If you want to see more information for more than one number, FAULT accepts up to ten error numbers and display the results.

```
2> FAULT 213 218 215
Fault 213: disk not validated
Fault 218: device (or volume) not mounted
Fault 225: not a valid DOS disk
```

What time is it, REALLY?

If you are one of those unfortunate people who owns a system without a battery backed-up clock, and you still use Preferences (as the generic startup-sequence commands you) to set the system date and time, you will be happy to know there is an easier way. The DATE command may be used to set the date and time and display it. If entered with no parameters, the DATE command returns the current system date and time.

```
2> DATE
Thursday 18-Feb-1988 21:32:16
```

To set the date, enter it as DD-MMM-YY. (The leading zeroes for DD may be left off.) You can use a day name (Tuesday) or YESTERDAY as well.

```
2> DATE 12-Jul-88
```

To set the time, enter it as HH:MM:SS. The following formats are valid:

```
2> DATE 10:      <— Sets the minutes and seconds to 00
2> DATE 13:42    <— Sets the seconds to 00
2> DATE 18:22:35
```

To write the date to a file, use the TO option.

```
DATE TO sys:lastboot
```

Although we have covered most AmigaDOS commands used in the CLI, we have barely scratched the surface of what can be done from the CLI. Next time, I'll introduce some commands and utilities that several bright programmers have given to the Amiga community. We'll also look at ways to get around some of the current limitations of AmigaDOS. Although most of us have read or heard about the forthcoming 1.3 upgrade, until Commodore makes it official and distributes it to the rest of us, we must find our own alternatives. In fact, some of the commands I'll be covering may be included with Workbench 1.3.

Please feel free to send any questions about using CLI commands, both AmigaDOS and Public Domain, to me care of this magazine. This column is designed to help users gain the most from the CLI and AmigaDOS. If you have trouble understanding any of what has been written in the past months, PLEASE let me know. I know many may still be fuzzy about batch files and the many features offered there. I intend to cover batch operations in more detail in the near future, so don't give up.

•AC•

F-Basic, from Delphi Noetic Systems, Inc.

A BASIC compiler that exploits the Amiga's power but not its own.

by Patrick Quaid

BASIC was originally designed as a programming language in which all lines were numbered, variables had one or two significant characters, and IF statements spanned only one line. There were no WHILE or REPEAT loops, and you could not write a program without scads of GOTOs. Power was an ON-GOSUB statement, and luxury was an automatic line renumbering utility.

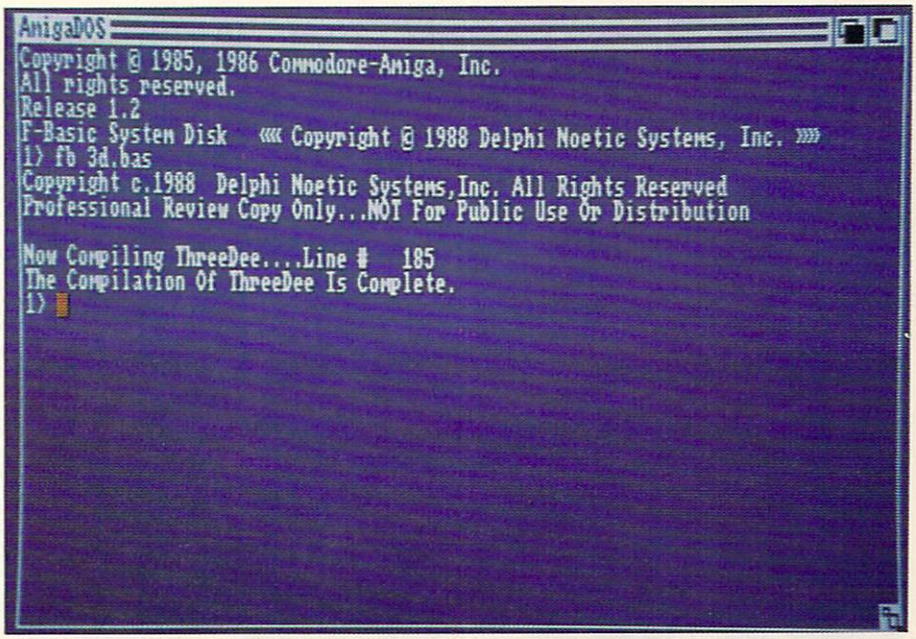
Doesn't sound much like AmigaBASIC, does it? Over the last few years, BASIC implementations have improved considerably. Gone are many of BASIC's limitations. In their place, more often than not, is Pascal. The more advanced a version of BASIC becomes, the more it begins to resemble Pascal.

F-Basic, a new BASIC compiler for the Amiga, progresses along this popular path. In fact, F-Basic has much more in common with Pascal than it does with traditional versions of BASIC. It is not compatible with AmigaBASIC, although virtually all programming techniques used in AmigaBASIC are possible in F-Basic.

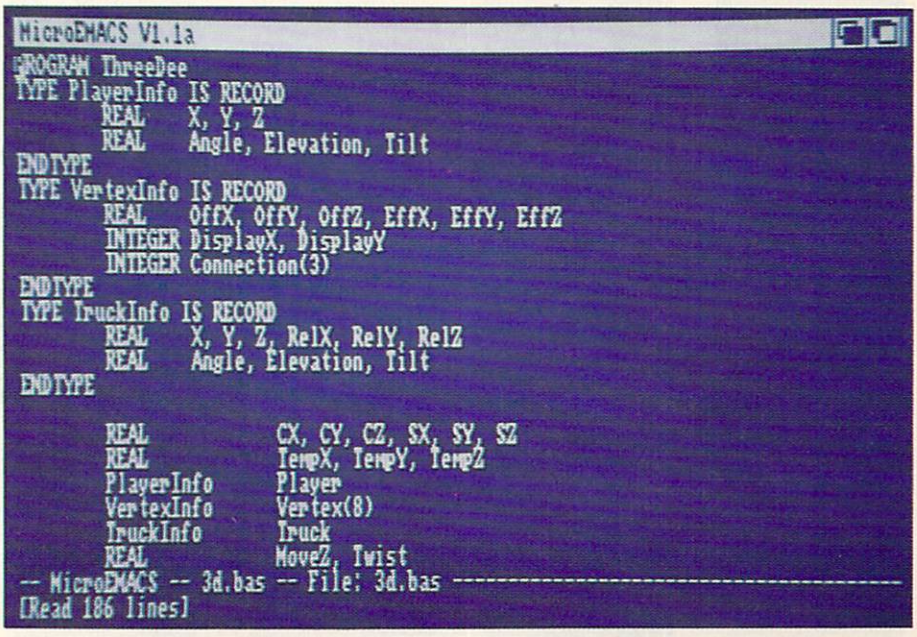
F-Basic stresses speed. The compiler is exceptionally fast, and produces speedy code. It is not surprising that F-Basic programs are much faster than AmigaBASIC programs; almost all compiled programs are faster than interpreted ones. What is surprising is that F-Basic's speed is comparable to just about any other high level language. For real arithmetic, F-Basic might be the fastest.

AmigaBASIC programmers face an almost entirely new language and programming environment in F-Basic. First of all, F-

(continued on page 59)



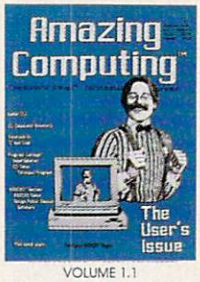
This screen shows the minimal development cycle.



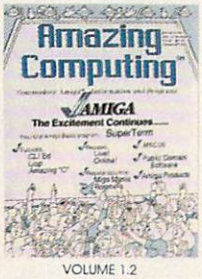
MicroEMACS editing, one of FBasic's demonstration programs.

Amazing COMPUTING™

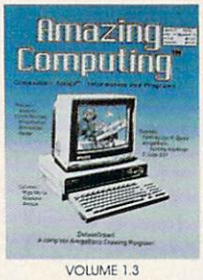
• E X P A N D I N G R E F E R E N C E •



VOLUME 1.1



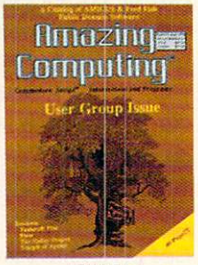
VOLUME 1.2



VOLUME 1.3

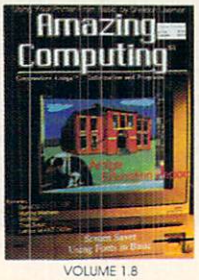
VOLUME 1.4

VOLUME 1.5

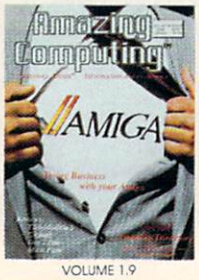


VOLUME 1.6

VOLUME 1.7



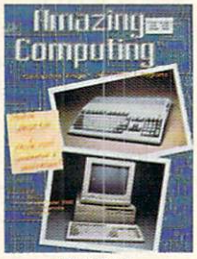
VOLUME 1.8



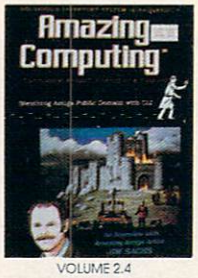
VOLUME 1.9

VOLUME 2.1

VOLUME 2.2

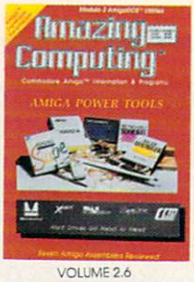


VOLUME 2.3



VOLUME 2.4

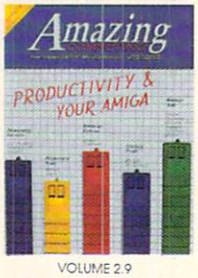
VOLUME 2.5



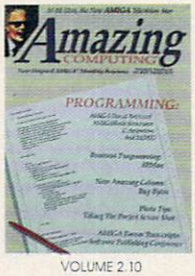
VOLUME 2.6

VOLUME 2.7

VOLUME 2.8



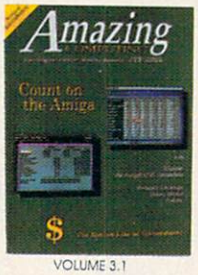
VOLUME 2.9



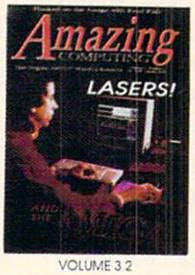
VOLUME 2.10

VOLUME 2.11

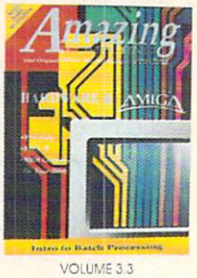
VOLUME 2.12



VOLUME 3.1



VOLUME 3.2



VOLUME 3.3

VOLUME 3.4

VOLUME 3.5

VOLUME 3.6

VOLUME 3.7

VOLUME 3.8

VOLUME 3.9

Your Resource to the Commodore Amiga

Expanding reference is not just an empty promise. The pages of Amazing Computing™ are filled with articles on technical operations and procedures, basic use, and just-plain-fun. The growing library of Amazing Computing's Back Issues contains articles ranging from building your own IBM Disk controller, to setting up your own startup sequence. Amazing Computing™ has repeatedly been the first magazine to offer the Amiga users solid, in depth reviews and hands on articles for their machines.

From the Beginning

Since February 1986, Amazing Computing™ has been providing users with complete information for their Amigas. This store house of programs and information is still available through our back issues. From the Premiere issue to the present, there are insights into the Amiga any user will find useful. AC was the first magazine to document CLI, tell its readers how to connect a 5 1/4 IBM drive, describe a 1 meg upgrade hardware project for the A1000, and many more. Please read the list of topics AC has covered below to find the information you have been missing.

Back Issues are $5.00 US, $6.00 Canada and Mexico, $7.00 Foreign Surface

All payments must be made by check or money order in U.S. funds drawn on a U.S. Bank.

Limited Supply

Unfortunately, nothing lasts forever, and the availability of some of our Back Issues is definitely limited. Complete your Amazing Computing™ library today, while these issues are still available, by completing the order form in the back of this issue.

Volume 1 Number 1 Premiere 1986

Super Spheres By Kelly Kauffman An ABasic Graphics prog.
Date Virus By J Foust A disease may attack your Amiga!
EZ-Term by Kelly Kauffman An ABasic Terminal program
Miga Manla by P. Kivolowitz Programming fixes & mouse care
Inside CLI by G. Musser a guided insight into the AmigaDos™
CLI Summary by G. Musser Jr. A list of CLI commands
AmigaForum by B. Lubkin Visit Compuserve's Amiga SIG
Commodore Amiga Development Program by D. Hicks
Amiga Products A listing of present and expected products

Volume 1 Number 2 March 1986

Electronic Arts Comes Through A review of software from EA
Inside CLI: part two G. Musser Investigates CLI & ED
A Summary of ED Commands
Live! by Rich Miner A review of the Beta version of Live!
Online and the CTS Fabito 2424 ADH Modem by J. Foust
Superterm V 1.0 By K. Kauffman A term. prog. in Amiga Basic
A Workbench "More" Program by Rick Wirch
Amiga BBS numbers

Volume 1 Number 3 April 1986

Analyze! a review by Ernest Viverios
Reviews of Racter, Barataccas and Mindshadow
Forth! The first of our on-going tutorial
Deluxe Draw!! by R. Wirch An Amiga Basic art program
Amiga Basic, A beginners tutorial
Inside CLI: part 3 by George Musser George gives us PIPE

Volume 1 Number 4 May 1986

SkyFox and Articfox Reviewed
Build your own 5 1/4 Drive Connector By Ernest Viveiros
Amiga Basic Tips by Rich Wirch
Scrimper Part One by P. Kivolowitz prog to print Amiga screen
Microsoft CD ROM Conference by Jim O'Keane
Amiga BBS Numbers

Volume 1 Number 5 1986

The HSI to RGB Conversion Tool
by S. Pietrowicz Color manipulation in BASIC
AmigaNotes by Rick Rae The first of the Amiga music columns
Sidecar A First Look by John Foust A first "under the hood"
John Foust Talks with R. J. Mical at COMDEX™
How does Sidecar affect the Transformer
an interview with Douglas Wyman of Simile
The Commodore Layoffs by J. Foust A look Commodore "cuts"
Scrimper Part Two by Perry Kivolowitz
Marauder reviewed by Rick Wirch
Building Tools by Daniel Kary

Volume 1 Number 6 1986

Temple of Apshai Triology reviewd by Stephen Pietrowicz
The Halley Project: A Mission in our Solar System
reviewed by Stephen Pietrowicz
Flow: reviewed by Erv Bobo
Textcraft Plus a First Look by Joe Lowery
How to start your own Amiga User Group by William Simpson
Amiga User Groups
Mailing List by Kelly Kauffman a basic mail list program
Pointer Image Editor by Stephen Pietrowicz
Scrimper: part three by Perry Kivolowitz
Fun With the Amiga Disk Controller by Thom Sterling
Optimize Your AmigaBasic Programs for Speed by Pietrowicz

Volume 1 Number 7 1986

Aegis Draw: CAD comes to the Amiga by Kelly Adams
Try 3D by Jim Meadows an introduction to 3D graphics
Aegis Images/ Animator: a review by Erv Bobo
Deluxe Video Construction Set reviewed by Joe Lowery
Window requesters in Amiga Basic by Steve Michel
ROT by Colin French a 3D graphics editor
"I C What I Think" Ron Peterson with a few C graphic progs
Your Menu Sir! by B Catley program Amiga Basic menues
IFF Brush to AmigaBasic 'BOB' Basic editor by M Swinger
Linking C Programs with Assembler Routines
on the Amiga by Gerald Hull

Volume 1 Number 8 1986

The University Amiga By G.Gamble
Amiga's inroads at Washington State University
MicroEd a look at a one man army for the Amiga
MicroEd, The Lewis and Clark Expedition reviewed Frizelle
Scribble Version 2.0 a review
Computers in the Classroom by Robert Frizelle
Two for Study by Frizelle Discovery & TheTalking Coloring Book
True Basic reviewed by Brad Grier
Using your printer with the Amiga
Marble Madness reviewed by Stephen Pietrowicz
Using Fonts from AmigaBasic by Tim Jones
Screen SaVer by P. Kivolowitz A monitor protection prog. in C
Lattice MAKE Utility reviewed by Scott P. Evernden
A Tale of Three EMACS by Steve Poling
.bmap File Reader in Amiga Basic by T Jones

Volume 1 Number 9 1986

Instant Music Reviewed by Steve Pietrowicz
Mindwalker Reviewed by Richard Knepper
The Alegra Memory Board Reviewed by Rich Wirch
TxEd Reviewed by Jan and Cliff Kent
Amazing Directory A guide to the sources and resources
Amiga Developers A listing of Suppliers and Developers
Public Domain Catalog A listing of Amicus and Fred Fish PDS
Dos 2 Dos review R. Knepper
Transfer files from PC/MS-DOS and AmigaBasic
MaxiPlan review by Richard Knepper The Amiga Spreadsheet
Gizmoz by reviewed by Peter Wayner Amiga extras!
The Loan Information Program by Brian Catley
basic prog. to for your financial options
Starting Your Own Amiga Related Business by W. Simpson
Keep Track of Your Business Usage for Taxes by J. Kummer
The Absoft Amiga Fortran Compiler reviewed by R A. Reale
Using Fonts from AmigaBasic, Part Two by Tim Jones
68000 Macros on the Amiga by G. Hull Advance your ability.
TDI Modla-2 Amiga Compiler review by S Faiwisze

Volume 2 Number 1 1987

What Digi-View Is... Or, What Genlock Should Be! by J. Foust
AmigaBasic Default Colors by Bryan Catley
AmigaBasic Titles by Bryan Catley
A Public Domain Modula-2 System reviewed by Warren Block
One Drive Compile by Douglas Lovell
Using Lattice C with a single drive system
A Megabyte Without Megabucks by Chris Irving
An Internal Megabyte upgrade
Digi-View reviewed by Ed Jakober
Defender of the Crown reviewed by Keith Conforti
Leader Board reviewed by Chuck Raudonis
Roundhill Computer System's PANEL reviewed by Ray Lance
Digi-Paint.....by New Tek previewed by John Foust
Deluxe Paint II ....from Electronic Arts previewed by J. Foust

Volume 2 Number 2 1987

The Modem by Josph L. Rothman efforts of a BBS Sysop
MacroModem reviewed by Stephen R. Pietrowicz
GEMINI or "It takes two to Tango" by Jim Meadows
Gaming between machines
BBS-PCI reviewed by Stephen R. Pietrowicz
The Trouble with Xmodem by Joseph L. Rothman
The ACO Project....Graphic Teleconferencing on the Amiga
by S. R. Pietrowicz
Flight Simulator II....A Cros Country Tutorial by John Rafferty
A Disk Librarian in AmigaBASIC by John Kennan
Creating and Using Amiga Workbench Icons by C. Hansel
AmigaDOS version 1.2 by Clifford Kent
The Amazing MIDI Interface build your own by Richard Rae
AmigaDOS Operating System Calls and
Disk File Management by D. Haynie
Working with the Workbench by Louis A. Mamakos Prog in C

Volume 2 Number 3

The Amiga 2000™ by J Foust
A First look at the new, high end Amiga™
The Amiga 500™ by John Foust
A look at the new, low priced Amiga
An Analysis of the New Amiga PCs by J. Foust
Speculation on the New Amigas
Gemini Part II by Jim Meadows
The concluding article on two-player games
Subscripts and Superscripts in AmigaBASIC by Ivan C. Smith
The Winter Consumer Electronics Show by John Foust
AmigaTrix by W. Block Amiga™ shortcuts
Intuition Gadgets by Harriet Maybeck Tolly
A journey through gadget-land, using C
Shanghai reviewed by Keith M. Conforti
Chessmaster 2000 & Chessmate reviewedby Edwin V. Apel, Jr.
Zing! from Meridian Software reviewed by Ed Bercovitz
Forth! by Jon Bryan Get stereo sound into your Forth programs.
Assembly Language on the Amiga™ by Chris Martin
Roomers by theBandito Genlocks are finally shipping, & MORE!!!
AmigaNotes by R. Rae Hum Busters... "No stereo? Y not?...
The AMICUS Network by J. Foust
CES, user group issues and Amiga Expo"

Volume 2 Number 4 1987

Amazing Interviews Jim Sachs by S. Hull Amiga Artist
The Mouse That Got Restored by Jerry Hull and Bob Rhode
Sluething Public Domain Disks with CLI by John Foust
Highlights: the San Francisco Commodore Show by S Hull
Speaker Sessions: San Francisco Commodore Show H Tolly
Household Inventory System in AmigaBASIC™ by B Catley
Secrets of Screen Dumps by Natkun Okun
Using Function Keys with MicroEmacs by Greg Douglas
Amigatrix II by Warren Block More Amiga shortcuts
Basic Gadgets by Brian Catley Create gadget functions
Gridiron reviewed by K. Conforti Real football for the Amiga
Star Fleet I Version 2.1 reviewed by J. Tracy Amigain Space
The TIC reviewed by J. Foust Battery powered Clock Calendar
Metascope review by H. Tolly An easy-to-use debugger

Volume 2 Number 5 1987

The Perfect Sound Digitizer review by R. Battle
The Future Sound Digitizer by W. Block Applied Vision's SD
Forth! by J. Bryancomparing JForth amd Multi-Forth.
Basic Input by B. Catley AmigaBASIC input routine for use in all your programs.
Writing a SoundScape Module in C by T. Fay Programming with MIDI, Amiga and SoundScape by SoundScape author.

Volume 2 Number 5 1987 continued

Programming in 68000 Assembly Language by C. Martin
Continuing with Counters & Addressing Modes.
Using FutureSound with AmigaBASIC by J. Meadows
AmigaBASIC Programming utility with real, digitized STEREO
AmigaNotes Rich Rae reviews SoundScape Sound Sampler.
More AmigaNotes by R. Rae A furtherlook at Perfect Sound .
Waveform Workshop in AmigaBASIC by J. Shields edit & save waveform for use in other AmigaBASIC programs.
The Mimetics Pro MIDI Studio by Sullivan, Jeffery
A review of Mimetics' music editor/player.
Intuition Gadgets Part II by H. MaybeckTolly Boolean gadgets provide the user with an on/off user interface.

Volume 2 Number 6 1987

Forth! by J. Bryan Access resources in the ROM Kernal.
The Amazing Computing Hard Disk Review by J. Foust & S. Leemon In-depth looks at the C Ltd. Hard Drive, Microbotics' MAS-Drive20, Byte by Byte's PAL Jr., Supra's 4x4 Hard Drive and Xebec's 9720H Hard Drive. Also. a look at disk driver software currently under devlopment.

Modula-2 AmigaDOS™ Utilities by S. Faiwiszewsk A
Calls to AmigaDOS and the ROM kernal.
Amiga Expansion Peripheral by J. Foust
Explanation of Amiga expansion peripherals.
Amiga Technical Support by J. Foust
How and where to get Amiga tech support.
Goodbye Los Gatos by J. Foust Closing Los Gatos.
The Amicus Network by J. Foust West Coast Computer Faire.
Metacomco Shell and Toolkit by J. Foust A review
The Magic Sac by J. Foust Run Mac programs on your Amiga.
What You Should Know Before Choosing an Amiga 1000 Expansion Device by S. Grant
7 Assemblers for the Amiga by G. Hull Choose your assembler
Shakeup Replaces Top Management at Commodore by S. Hull
Peter J. Baczor"by S. Hull Manager at CBM gives an inside look
Logistix A review by Richard Knepper
Organize! by A review Richard Knepper database.
68000 Assembly Language Programming on the Amiga
by Chris Martin
Superbase Personal Relational Database by Ray McCabe
AmigaNotes by Rae, Richard A look at FutureSound
Commodore Shows the Amiga 2000 and 500 at the Boston Computer Society by H Maybeck Tolly

Volume 2, Number 7 1987

New Breed of Video Products by John Foust...
Very Vivid! by Tim Grantham...
Video and Your Amiga by Oran Sands III
Amigas & Weather Forecasting by Brenden Larson
A-Squared and the Live! Video Digitizer by John Foust.
Aegis Animator Scripts and Cel Animation by John Foust
Quality Video from a Quality Computer by Oran Sands III.
Is IFF Really a Standard? by John Foust..
Amazing Stories and the Amiga™ by John Foust.
All about Printer Drivers by Richard Bielak
Intuition Gadgets by Harriet Maybeck Tolley.
Deluxe Video 1.2 by Bob Eller
Pro Video CG1 by Oran Sands III.
Digi-View 2.0 Digitizer/Software by Jennifer M. Janik
Prism HAM Editor from Impulse by Jennifer M. Janik
Easyl drawing tablet by John Foust..
CSA's Turbo-Amiga Tower by Alfred Aburto
68000 Assembly Language by Chris Martin.

Volume 2, Number 8 1987

This month Amazing Computing™ focuses on entertainment packages for the Amiga. Amazing game reviews...

SDI, Earl Weaver Baseball, Portal, The Surgeon, Little Computer People, Sinbad, StarGlider, King's Quest I,II and III, Faery Tale Adventure, Ultima III, Facets of Adventure, Video Vegas and Bard's Tale.

Plus Amazing monthly columns... Amiga Notes, Roomers, Modula-2, 68000 Assembly Language and The Amicus Network.
Disk-2-Disk by Matthew Leeds
The ColorFonts Standard by John Foust
Skinny C Programs by Robert Riemersma, Jr.
Hidden Messages In Your Amiga™ by John Foust
The Consumer Electronics Show and Comdex by J Foust

Volume 2 Number 9 1987

Analyze 2.0 reviewed by Kim Schafer
Impact Business Graphics review by Chuck Raudonis
Microfiche Filer review by Harv Laser
Pagesetter review by Rick Wirch
Gizmoz Productivity Set 2.0 review by Bob Eller
Kickwork review by Harv Laser
Diga Telecommunications Package review by Steve Hull
Mouse Time and Timesaver review by John Foust
Insider Memory Expansion review by James O'Keane
Microbotics Starboard-2 review by S. Faiwiszewski
Leather Goddesss of Phobos
reviewed by Harriet Maybeck-Tolly
Lattice C Compiler Version 3.10 reviewed by Gary Sarff
Manx 3.4a Update reviewed by John Foust
AC-BASIC reviewed by Sheldon Leemon
AC-BASIC Compilier an alternative comparison by B Catley
Modula-2 Programming S Faiwiszewski
Raw Console Device Events
Directory Listings Under AmigaDOS by Dave Haynie
AmigaBASIC Patterns by Brian Catley
Programming with Soundscape
Todor Fay manipulate's samples
Bill Volk, Vice-President Aegis Development,
interviewed by Steve Hull
Jim Goodnow, Devekoper of Manx 'C' interview by Harriet M Tolly
Plus a great collection of monthly columns...

Volume 2 Number 10 1987

Max Headroom and the Amiga by John Foust
Taking the Perfect Screen Shot by Keith Conforti
Amiga Artist: Brian Williams by John Foust
Amiga Forum on Compuserve™... Software Publishing Conference Transcript by Richard Rae
All About Online Conferencing by Richard Rae
dBMAN reviewed by Clifford Kent
Amiga Pascal reviewed by Michael McNeil
AC-BASIC Compiler reviewed by Bryan Catley
68000 Assembly Language by Chris Martin
Amiga Programming:
Amiga BASIC Structures by Steve Michel
Quick and Dirty Bobs by Michael Swinger
Directory Listings Under Amiga-DOS, Part II by Dave Haynie
Fast File I/O with Modula-2 by Steve Faiwiszewski
Window I/O by Read Predmore
Plus a great collection of monthly columns...

Volume 2 Number 11 1987

Word Processors Rundown by Geoff Gamble
ProWrite, Scribble!, and WordPerfect compared
LPD Writer Review by Marion Deland
VizaWrite Review by Harv Laser
Aedit Review by Warren Block
WordPerfect Preview by Harv Laser
Jez San Interview by Ed Bercovitz—StarGlider author speaks!
Do-it-yourself Improvements to the Amiga Genlock
Digi-Paint Review by Harv Laser
Sculpt 3D Review by Steve Pietrowicz
Shadowgate Review by Linda Kaplan
TeleGames Review by Michael T. Cabral
Reason Preview: an intense grammar examination application
As I See It by Eddie Churchill Peeking at WordPerfect, Gizmoz V2.0 and Zing! Keys
AmigaNotes by R Rae 4 electronic music books
Modula-2 Programming by Steve Faiwiszewski devices, I/O, and the serial port
68000 Assembly Language by Chris Martin Chris walks through the display routines
The AMICUS Network by John Foust—Desktop Publishing, Seybold
C Animation Part II by Mike Swinger Animation Objects
BASIC Text by Brian Catley Pixel perfect text positioning
Soundscape Part III by Todor Fay VU Meter and more
Fun with Amiga Numbers by Alan Barnett
File Browser by Bryan Catley—Full Feature BASIC File Browsing
Plus a great collection of monthly columns...

Volume 2 Number 12 1987

The Ultimate Video Accessory by Larry White
The Sony Connection by Stewart Cobb
15-Puzzle in AmigaBASIC by Zoltan Szepsi
Life, Part I: The Beginning by Gerald Hull
The ultra-complex nine bit solution to the "Game of Life."
Amiga Virus! by John Foust
A new Amiga virus has surfaced. Please check your system.
CLI Arguments in C by Paul Castonguay
MIDI Interface Adapter by Barry Massoni
Amiga 1000-style MIDI interfaces can fit A2000s or 500s
Modula-2 by Steve Faiwiszewski
First in a series, a command line calculator in Modula-2.
AmigaNotes by Richard Rae
The audio changes made in the Amiga 500 and 2000.
Animation for C Rookies: Part III by M. Swinger
tackling double-buffering.
The Big Picture by Warren Ring
Amiga™ Assembly language programming for the brave!
Karate Kid Review by Stephen R. Pietriowicz
GO! 64 review by John Foust, James O'Keane, and Rick Wirch
Three C-64 experts investigate a new Amiga 64 emulator.
A-Talk-Plus Review by Brendan Larson
"Full-fledged terminal program" & Tektronics capabilities
Calligrapher Review by John Foust
Animator: Apprentice Review by John Foust
Playing Dynamic Drums on the Amiga by David N. Blank
WordPerfect Review by Steve Hul
Insider/Kwikstart Review by Ernest P. Viveiros Sr
RAM & ROM expansion: Comments and installation tips.
Bug Bytes by John Steiner
Forth! by Jon Bryan
DumpRPort utility for your Multi-Forth toolbox.
As I See It by Eddie Churchill
An offbeat look on Digi-Paint, Portal, and Videoscape 3D.
The Amicus Network by John Foust
The Commodore Show and AmiExpo: New York!

Volume 3 Number 1 1988

AmigaNotes by Richard Rae
Digital music generation on tha Amiga .
C Animation Part IV by Michael Swinger
Just when you thought it was safe to go back in the C waters...
Forth by John Bryan
Sorting out CHIP and FAST memory on the Amiga.
The Big Picture by Warren Ring Daring assembler language programming; CLI system calls and manipulating disk files.
Bug Bytes by John Steiner
Roomers by The Bandito
Amiga Dos1.3? 80386-based BridgeBoard for the A2000? More!
As I See It by Eddie Churchill
Opinions, obsevations, & the birth of a new software generation.
68000 ASssembly Langueage Programming by Chris Martin
"Create a multi-color screen without using Intuition routines!"
Modula-2 Programming by Steve Faiwiszewski
A new contender bursts on to the modula-2 scene!
Amicus Network Special Report: Fall COMDEX by J. Foust
Commodore at COMDEX and new products
The ultimate Video Accessory: Part II by Larry White
Life: Part II by Gerald Hull
"A detailed look at efficient use of the Amiga bliter."
FormatMaster: Professional Disk Formatting Engine byC.Mann
Put Batch language to work on the drudgery of disk formatting .
BSpread by Brian Catley
A full featured AmigaBASIC spreadsheet you can program!
AmigaForum Transcript ed. by Richard Rae
Zoom in on Commodore/ Amiga's Dave Haynie.
Haicalc Review by Chuck Raudonis
"A straightforward, easy to use, functional spreadsheet."
VIP Professional Review by Suzanne Mitchell
Easy stock porfolio management on the Amiga.
Money Mentor Review by Stephen Kemp
A personal finance system beyond your checkbook.
Investor's Advantage Review by Richard Knepper
plus "Poor Man's Guide to the Stock Market."

Volume 3 Number 2 1988

Laser Light Shows with the Amiga by Patrick Murphy
Lasers and the Amiga: A Dazzling Tandem
The Ultimate Video Accessory: PartIII by Larry White
Take the final stepstoward designing your own videos.
Our First Desktop Video by Larry White
Step-by -step guide to organizing & presenting your Amiga video.
Hooked on the Amiga with Fred Fish by Ed Berkovitz
Inside views from the man behind all those "Fish" disks.
Photo Quality Reproduction with the Amiga and Digi-View by Stephen Lebans
Balancing your Checkbook with WordPerfect Macros by S.Hull
Hand your checkbook worries over to the Amiga.
More Basic Text by Bryan Catley easier text on an Amiga screen
Life: Part III by Gerald Hull
Sries winds up with famed nine-bit calculation & source to LIFER.
Solutions to Linear Algebra through Matrix Computations by Roobert Ellis
Simplify matrix algebra with basic operations & routines.
Roomers by Bandito Amiga 3000, Virus news,& Laser Toaster
Bug Bytes by John Steiner
Modula-2 Programming by Steve Faiwiszewski
Catching up with Calc-a source follow-up.
68000 Assembler Language Programming by Chris Martin
Graphics- Part II of Assemgram.
Arazok's Tomb by Kenneth E. Schaefer
"A terrifyig adventure into the world of the occult."
AiRT by Steve Faiwiszewski
An innovative icon-basede programming language.
Forms in Flight by Steve Pietrowicz
Render and Animate objects in 3D!
Silicon Dreams and the Jewel of Darkness by K E. Schaefer
Leisure suit Larry by Kenneth E. Schaefer
Two New Entries From Microbiotics by John Foust
M501 Expansion & Starboard II MultiFunctionboard.
Mindlight 7 and People Meter by John Foust
Phantasie
Ken E. Schaefer with the Amazing Phantasie Character Editor.

Volume 3 Number 3 1988

Take Five! by Steve Hull
Blast your frustrations in these five Amiga games.
Desktop Video,Part IV by Larry White
Put all the pieces together-the desktop video commercial.
The Hidden Power of CLI Batch File Processing by J. Rothman
Make your Amiga easier to use with CLI Batch files.
A Conference with Eric Graham edited by John Foust
The mastermind behind Sculpt 3D and Animate 3D.
Perry Kivolowitz Interviewed by Ed Bercovitz
Amiga insights from a major developer and personality.
Jean "Moebius" Giraud Interviewed by Edward L. Fadigan
Avant-garde art comes to the Amiga-in dazzling form.
PAL Help by Perry Kivolowitz
All the help you need for a A1000 expansion reliability.
Boolean Function Minimization by Steven M. Hart
A useful digital design tool in AmigaBASIC.
Amiga Serial Port and Midi Compatibility for Your A2000!
by Lynn Ritter and Gary Rentz
Add an A1000-style serial port to your A2000!
Electric Network Solutions the Matrix Way by Robert Ellis
Engineers! Practicle routines for using matrix algebra.
The A.M.U.G. BBS List compiled
by Joe Rothman, Chet Solace, and Dorothy Dean 100 514 BBS phone numbers in the U.S. and Canada.
FACC II reviewed by Graham Kinsey
Put a firecracker under your floppy drives.
Uninvited reviewed by Kenneth E. Schaefer
When was the last time a game scared you to death?
Flow reviewed by Pamela Rothman
Turn your brainstorms into mental works of art.
Benchmark Modula-2 Compiler reviewed by Richie Bielak
Program development that beats Pascal to the punch.
Modula-2 Programming by Steve Faiwiszewski
The gameport device and simple sprites in action.
AmigaNotes by Richard Rae
A1000! Create a software-switchable output filter.
Roomers by The Bandito
Inside AmiExpo ... Kickstart 1.4 .. Commodores 'R' Us?
The Big Picture by Warren Ring--Unwiedly arguments to system calls? Discover the Unified Field Theory!!
Plus a great collection of monthly columns...

Volume 3 Number 4 1988

Highlights from AmiExpo, Los Angeles by Steve Hull
The Amiga shows off its best in the West.
Writing a SoundScape Patch Librarian by Todor Fay
Get your hands dirty working within the System Exclusive.
Upgrade Your A1000 to A500/2000 Audio Power—by H.Bassen
Modifications to help your A1000 make sweet music, too!
Amiga Audio Guide
Descriptive listing of all Amiga audio products.
Gels in Multi-Forth by John Bushakra
Push Gels to the limit with these programming tools.
Macrobatics by Patrick J. Horgan Ease the trauma of assembly language programming.
Amiga Audio Sources The folks behind all those audio products.
Take Five! by Steve Hull
Four lightning-paced titles to slash boredom.
Amiga Notes by Rick Rae
Confounded by sound? Take a basic tour of Amiga audio.
The Ultimate Video Accesory, Part V by Larry White
Let's add some flash to our video.
Bug Bytes by John Steiner
The Big Picture by Warren Ring
Part II of the eye-opening Unified Field Theory.
Roomers by The Bandito
Hardware hijinx ... Toasted video ... the dream Amiga ... and more!
In the Public Domain by C.W. Flatte
C.W. has hooked the latest Fish disks—here's an inside look.
Time Bandit review by Keith Conforti
A whole video arcade wrapped up in one game!
AudioMaster review by Brendan Larson
Friendly digitizing software that samples in real-time.
Music Mouse review by J Henry Lowengard
Making music without lifting a finger from the mouse.
Amiga-Tax/Canadian Version review by Ed Bercovitz
A Canadian income tax planning, preparation, & analysis package.
SAM BASIC review by Bryan Catley
A new BASIC which exploits even more unique Amiga features.

Volume 3 Number 5 1988

Interactive Startup Sequence by Udo Pernisz
The Command Line part 1by Rich Falconburg
AmigaTrix III by Warren Block—Tips and tidbits to ease Amiga life
Amiga Product Guide: Hardware Edition
Proletariat Programming by P Quaid—Public domain compilers
The Companion by Paul Gosselin
The Amiga's Event Handling capability.
MindLight 7 reviewed by David N. Blank
Psychedelic fad of the 70's updated for the Amiga.
VideoScape 3-D 2.0 reviewed by David Hopkins
Extend reviewed by Bryan D. Catley—An AmigaBASIC extension
AssemPro reviewed by Stephen Kemp
Opening a door to assembly language programming.
APL.68000 reviewed by Roger Nelson
Book Reviews by Richard Grace—Three "C" programming texts.
CBTREE reviewed by Michael Listman
A tidy collection of functions to aid the C programmer.
The Big Picture by Warren Ring
The three-part Unified Field Theory winds up
Modula-2 by Steve Faiwiszewski
Termination modules for Benchmark and TDI compilers.
68000 Assembly Language by Chris Martin
Peeling away the complication of display routines.
Plus a great collection of monthly columns...

Volume 3 Number 6 1988

Bear Time Reviewed by Steve Carter
What makes this inexpensive A1000 battery-backed clock tick?
Acquisition Reviewed by David N. Blank
A look inside the latest release of a powerful relational database.
Butcher 2.0 Reviewed by Gerald Hull
A tidy collection of diverse image processing utilities.
Reassigning Workbench Disks by John Kennan
Endless disk swapping comes to a merciful end.
Product Guide: Software Tools Edition
A listing of all the products you need to put your Amiga to work.
An IFF Reader in Multi-Forth by Warren Block
Create an easy to use IFF reader in Multi-Forth.
Basic Directory Service Program by Bryan Catley
A programming alternative to the GimmeeZeroZero windows.
C Notes from the C Group by Stephen Kemp
A beginner's guide to the power of C programming.
An Amiga Forum Conference with Jim Mackraz
The Amiga market as seen by the "Stepfather of Intuition."
Son of Seven Assemblers Reviewed by Gerald Hull
A ocmparative battle between seven native-code assemblers.
The 1988 Commodore Amiga Developers Conference
A look inside the conferences held in Washington, D.C.
Amiga Working Groups by Perry Kivolowitz and Eric Lavitsky
An outline of the innovative Amiga Working Groups concept.
The Command Line by Rich Falconburg
Exploring the multi-talented LIST command.
Plus a great collection of monthly columns...

Volume 3 Number 7 1988

Look, Up On the Screen, It's an Ami... It's a Pro... It's SuperGen
reviewed by Larry White—Genlock comparisons
An Interview with "Anim Man," Gary Bonham by B. Larson
An animated conversation with the man behind the format.
The Amiga at Spring COMDEX in Atlanta by Ed Bercovitz
The Amiga and its third party partners wow the South.
Amiga Product Guide: Video/Graphics Edition
Thirteen pages devoted to the Amiga's dazzling strong suit.
The Developing Amiga by Steve Pietrowicz
Developers' notes: PD vs. shareware vs. freely distributableetc.
Roll Those Presses! by Barney Schwartz
Welcome to the dandy, demanding world of desktop publishing!
Linked Lists in C by William E. Gammill
Put dynamic memory to work with linked lists.
FrameGrabber Preview by Oran Sands
Capturing an image can now be as fast as punching a single key!
A First Look at Interchange reviewed by David Hopkins
Bridge the gap between those incompatible animation packages.
Perfect Vision reviewed by Bryan Catley
Capture, digitize and save pictures from any video source.
ProWrite 2.0 Review reviewed by Pamela Rothman
A graphic word processor specializing in efficient editing.
Doug's Math Aquarium: The Art of Mathematics by Richie Bielak
Solving equations was never this much fun!
Bear Products MegaRex II Expansion RAM by Steve Carte
An inexpensive way to pump your Amiga up to 2MB.
The Command Line by Rich Falconburg
Amiga Notes by Rick Rae
The Other Guys' Synthia: A digital synthesizer features won't stop.
C Notes from the C Group by Stephen Kemp
Weathering the unknown "C" of basic object and data types.
Plus a great collection of monthly columns...

Volume 3 Number 8 1988

The Command Line by Rich Falconburg
The journey into the CLI continues.
The Developing Amiga by Stephen R. Pietrowicz
A gaggle of great programming tools.
Modula-2 Programming by Steve Faiwiszewski
Libraries and the FFP and IEE Math Routines.
C Notes from the C Group by Stephen Kemp
Arrays and pointers unmasked.
Dark Castle reviewed by Keith Conforti—The Black Knight lurks
Ports of Call reviewed by Julie Landry
Leatherneck reviewed by Michael Creeden-Rambo's not so tough!
Capone reviewed by Joyce and Robby Hicks- Light Guns blaze
Casino Fever reviewed by Michael T. Cabral—Vegas on Amiga
Ferrari reviewed by Jeffery Scott Hall—Start your engine
Arkanoid reviewed by Graham Kinsey—"blockbuster"
Ebonstar by Keith Conforti —black hole treking.
Deluxe Productionsreviewed by Harv Laser—Video wizardry
Game Pizazz by Jeffery Scott Hall— Register your questions here.
TrackMouse by Darryl Joyce
Convert a standard Atari trackball into a peppy Amiga TrackMouse.
Amiga Interface for Blind Users reviewed by Carl W. Mann
An ingenious interface that opens the Amiga to even more users!
Video in the Sunshine State reviewed by Stephen R. Pietrowicz
RGB Video Creations hosts a video unveiling!
Amiga Product Guide: Games Edition
Tumblin' Tots by David Ashley
An assemply language game you can program.
Plus a great collection of monthly columns...

Volume 3 Number 9 1988

The Kideo Tapes by John Dandurand
A Georgia elementary school puts desktop video to work.
Speeding Up Your System by Tony Preston
Fuel inject your system with floppy disk caching
Amiga Product Guide: Education Edition
Everything you need to send your Amiga to the head of the class.
Computer Aided Instruction by Paul Castonguay
Authoring system in AmigaBASIC.
Gels in Multi-Forth, Part II: Screenplay by John Bushakra
Make the IFF converter from Part I easy to use-gadgets, menus,etc.
AmiExpo Midwest '88 by Michael T. Cabral
After taking the coasts by storm, the Amiga wows Chicago
Intellitype by Harv Laser—Learning to type made easy...and *fun?*
Shakespeare by Barney Schwartz—Desktop publishing in full color.
XSpecs 3D by Steve Hull—A new dimension in Amiga graphics.
AmigaNotes by Richard Rae—How IFF sound samples are stored?
Take Five! by Steve Hull—Beat the back-to-school blues !
The Command Line by Rich Falconburg—continuing tour of CLI.
Hot on the Shelves by Michael T. Cabral & Michael Creeden
What do you get when you combine intense war strategy with a monochrome monitor and desktop presentation? Check it out.
Bug Bytes by John Steiner
C Notes from the C Group by Stephen Kemp
Operators, expressions, and statements in C uncovered.
Roomers by The Bandito
Can an Apple IIgs Plus a day keep the Amiga away?

To be continued.....

(FBasic, continued from page 55)
Basic does not have an integrated editor. Programs are written with a normal ASCII text editor, like ED or Micro-EMACS, then compiled with a separate program. This might seem inconvenient at first, but most compiled languages work this way. Actually, it's nice to be able to use your favorite editor in place of the default. Anyone who uses AmigaBASIC can appreciate this feature.

Variables and routine names are case sensitive, which might irritate some programmers. As in Pascal, all F-Basic programs begin with "PROGRAM Name" and end with "END." Variables must be declared by type at the beginning of the program, and all strings have a fixed maximum length. These requirements are common to other compiled languages, but are definitely new to BASIC.

F-Basic has all the familiar decision and looping structures, such as IF/THEN blocks (with ELSE and ELSEIF parts), CASE-type statements, and REPEAT, WHILE and FOR loops. These make the the GOTO statement unnecessary. (Nonetheless, the GOTO is included.)

Aside from speed, F-Basic's second claim to fame is that it includes record structures and pointer variables. These too are common in other programming languages, and although an absolute requirement for serious Amiga programming, they are still missing from AmigaBASIC. Records and pointers allow F-Basic programs to interact intelligently with Amiga ROM routines without forcing arrays and integer variables to do the job. They are two powerful features made possible by the overhead of F-Basic and other compiled languages.

F-Basic also provides direct access to Amiga system routines. More work is involved than what's required for similar calls in AmigaBASIC, but at least .bmap files are not required. The interface to the libraries is defined in a text file called FastSysLib, which is accessed only during compilation. This file's format is not explained in the F-Basic manual, however, so adding support for other libraries is not immediately possible.

In addition to being very fast, F-Basic also supports real aritmetic with an array of functions.. F-Basic has all the common trigonometric functions, like sine, cosine and tangent, as well as inverses, hyperbolic functions, and co-functions. In most BASICs, only a few of these are provided, and the others must be derived from these classics. F-Basic also has random functions with both real and integer results, as well as normal exponential functions.

F-Basic also shines in its handling of strings. There are routines to sort, fill, and search for strings, convert upper to lower case and back, and several others. Unfortunately, strings in F-Basic are fundamentally flawed, so this icing covers a lousy cake. Most languages, from Assembly to C to Modula-2 to AmigaBASIC, end strings with a zero byte. The Amiga system software expects them this way. F-Basic does not use this method, and, in fact, has no way of determining the correct length of a string. Instead, it uses the position of the last non-blank character as the end of the string, so trailing spaces are always insignificant. In addition, if a long string is assigned a shorter string, the new length is not recorded. In other words, if Str holds the value "A Big String," and later the line Str= "Short" is executed, the result is "Short String." F-Basic expects the string to be cleared before the assignment, and provides simple methods for doing so, but this lack of accuracy severely handicaps strings in F-Basic.

F-Basic strings also run into problems with comparisons. In order for strings to be alphabetically compared, they must be the same length. Otherwise, comparisons indicate that they are, all at once, unequal, greater than, and less than each other, regardless of what they actually contain. Since one of BASIC's traditional strengths is handling strings, this poor implementation is especially disappointing. Even C, with no real text type, does not impose so many restrictions on string manipulation.

On the plus side, F-Basic has great pattern matching capabilities. Based on concepts from SNOBOL4, the routines use templates to match strings in complex, non-linear ways. These routines definitely are a language in themselves, and the manual devotes plenty of space to them. The routines make implementing the common sentence parsing techniques of adventure games almost trivial. If you don't learn to use them, F-Basic provides a full slate of familiar string searching procedures.

F-Basic's unfamiliar requirements and numerous built-in routines are explained in a nearly 200-page manual. For an initial version, it is quite complete and informative, although, in this age of desktop publishing, the monotonous typewriter-like font is tiresome. None of my questions were left unanswered, and as questions from other users find their way into it, the manual will inevitably improve. It has a complete table of contents, but the index could use substantial expansion, and a table of F-Basic's functions ambles through thirteen pages and lacks syntax information.

To its generally sound base, F-Basic adds high level support for Amiga-specific capabilities, such as windows, screens, menus, speech, and graphics. For several reasons, F-Basic's performance in this area is vastly superior to AmigaBASIC's. For one thing, F-Basic does not unnecessarily encumber windows. It uses simple Intuition windows, so AmigaBASIC's annoying window-moving lag is gone. F-Basic's high level support also integrates smoothly with low-level system routines. The high level commands that open screens and windows, for example, return pointers to the appropriate Intuition records. These records, in turn, are required by many system-level screen and window manipulation routines. This extra consideration illustrates the degree to which F-Basic expects the programmer to operate at various levels.

F-Basic handles other areas of Amiga programming with similar cleverness. To process input events, for example, a programmer defines routines to be executed whenever a particular type of event occurs. Separate blocks can be defined to handle a key stroke, a menu selection, a single or double mouse click, or selection of a window's close gadget.

Within the block, pertinent information (like the mouse position or menu number) is made available. When the block is finished, execution resumes where it left off, so events are treated like high level interrupts. There are very few restrictions on what can occur in one of these blocks. You can, for instance, have a CLOSE_WINDOW block not only close the window, but also end the program with no side effects.

AmigaBASIC's GET and PUT commands are very easy to use, but too slow for animated graphics. F-Basic solves this problem by introducing a series of BLOCK commands which can grab a rectangle from a window, put it back down in any window, or transfer the information in and out of arrays. This is all done very quickly, since the function uses the blitter whenever possible. When it redraws the block, all the blitter's Boolean operations are available. The functions, although powerful and lightning-fast, are as easy to use as AmigaBASIC's GET and PUT.

F-Basic also supplies high-level access to most Amiga graphics capabilities. All graphics library drawing functions are supported. Other functions control the cursor location, text style (italic, bold, underlined, etc.), and color combinations. This bounty, unfortunately, does not include a simple method for clearing the window; the routines here simply clear the area where characters can be printed, leaving a cluttered border. Generally, however, the graphics functions are complete and flexible. Of course, the system routines are always available, so every Amiga faculty should be available in some form.

F-Basic does not yet provide high-level mouse or joystick routines, but these too can be handled by system calls. Apparently the revision will provide random file support that this version also lacks.

Unlike AmigaBASIC, F-Basic tries to provide the features (like the records and pointers discussed above) that make serious program development possible. Other examples of this effort include pre-initialized variables, constants, and local variables. Like Pascal, F-Basic also recognizes the difference between a procedure and a function (a function is expected to return a value). Even function results are returned in Pascal form. These abilities are not absolutely necessary, but they significantly reduce logic errors in large programs.

F-Basic's access to the microprocessor is another sign of serious development support. The 68000's registers can be read and set directly with a syntax that makes them as easy to use as normal variables. The F-Basic manual discusses when the registers are available, and what they are used for at various times. For extra speed, the language uses four registers to hold variables. Rather than forcing the compiler to estimate which variables are best suited for this, F-Basic simply uses the first two or four variables declared, depending on size. Knowing this, programmers can control which variables are used.

This is just the beginning of processor-level control. F-Basic has operators which use the 68000 left and right shift commands to multiply and divide a variable by a power of two. If you need to double or halve a number, these are much faster than normal math functions. F-Basic also supports a compiler directive that speeds up normal multiplications and divisions involving small numbers. Optimizing compilers tend to take care of this stuff automatically, but the fact that F-Basic leaves it up to the programmer means optimization is always undertaken where appropriate. Guidelines for these options are given in the manual.

Although many programmers would rather not admit it, debuggers are another important aspect of serious software development. Interpreted BASICs typically have good debugging support with ways to trace programs, examine variables during a run, etc.

Debuggers provide similar support for compiled languages, often with many additions. Source level debuggers, which interact with a program using its source code rather than the assembly language produced by the compiler, are often the easiest to use. Delphi Noetic reportedly has such a debugger in the works. It should be available by the time this article is published, and at a very reasonable $50 for new purchasers, or $25 for current F-Basic owners. This extension of F-Basic's environment is a very important aspect of the language's overall value, and shows the support the product will apparently receive.

Obviously much effort was put toward making F-Basic a viable Amiga development language. The system shoots itself in the foot, however, by not including a linker. This means that every F-Basic program needs an additional file, called FastLib, to run. FastLib is F-Basic's run-time library—a group of routines that together provide the language's features. Most compiled languages do not use a separate file, since essential parts of the run-time library are attached directly to the program by the linker.

This tag-along program introduces several problems. The most obvious is that a particular disk must always be available to the program. When an F-Basic program first runs, it looks for the file in the current directory or the SYS:s directory. To lessen the inconvenience of FastLib, the program should look in an assigned device like S: (or, better yet, LIBS:), so the user can have more control over the environment. The only way to store FastLib on the RAM: disk using this version of F-Basic is to make the RAM: disk also the SYS: disk, which is not feasible for many Amiga owners.

Benchmark Results

For the Calc Error, zero would be perfect, so smaller absolute values are better. The other values are run times measured in seconds.

	Calc Time	Calc Error	Sieve	PSET
F-Basic	1.28	-9.313225 E-010	2.98	11.90
AmigaBASIC				
single	15.18	-5.960464 E-08	524.34	38.77
double	20.02	-1.110223 E-016		
AC/BASIC	4.89	-1.79 E-07	114.6	21.96
TDI Modula-2				
single	2.78	-5.960464 E-08	6.12	6.20

These are the F-Basic versions of the benchmark programs. The AmigaBASIC and AC/BASIC versions are listed in a review of AC/BASIC in AC V2.9. I've omitted the timing subroutine from each program, but remarked where the time was measured.

CALC.BAS

```
PROGRAM Calc
CONSTANT A= 2.71818, B= 3.14159, NR=
5000
REAL C
INTEGER i
REM The DATA statement pre-initializes
variables
REM Thus when the program loads, C will
already be 1.0
DATA(C,1.0)

REM Note the start time

FOR i= 1 TO NR
   C= C * A
   C= C * B
   C= C / A
   C= C / B
NEXT

REM Note end time, and print difference.
PRINT "Error is ",C - 1.0
END
```

SIEVE.BAS

```
PROGRAM Sieve
CONSTANT Size= 8190
INTEGER i, prime, k, count, iter
BYTE flags(Size + 1)

REM Note initial time
FOR iter= 1 TO 10
   count= 0
REM the following sets the entire array to
a single value
   flags= TRUE
   FOR i= 1 TO Size+1
      IF flags(i) THEN
         prime= i+i+3
         k= k+prime
         WHILE k<= Size+1
            flags(k)= FALSE
            INC(k,prime)
         ENDWHILE
         INC(count)
      ENDIF
   NEXT
NEXT
REM Note final time, and print difference
END
```

PSET.BAS

```
PROGRAM PSET
INTEGER i,j

i= WINDOW #1 (0,0,640,200,50,50,-1,-1,-1,-
1,23,@"PSET Test",-1)
REM Note initial time

FOR i= 50 TO 250
   FOR j= 50 TO 150
      COLOR_POINT (i,j)
   NEXT
NEXT

REM Note end time and print difference
WINDOW_CLOSE #1

END
```

A second problem caused by the run-time library is that although only one copy of the file need be on a disk, each program loads its own complete version. At 35K per copy, this operation wastes a lot of memory, especially for small programs that use only a few of the library's features anyway. F-Basic should include a linker that selectively attaches required routines to a program (a feature provided by most compiled languages). Omitting the linking stage speeds up the development process, but this is not a valid reason for leaving out an important tool. The nuisance of the library is more obvious for small utility programs, but since the language supports smaller programs so well, the run-time environment should also provide the support.

A second major limitation of F-Basic programs is that they must be run from a CLI. Even if the program does not attempt input or output, or opens a window for its first instruction, it crashes the system if icon-invoked. This is another problem that evidently will be repaired in the pending release.

To compare F-Basic's performance to AC/BASIC's performance, I used the benchmarks and AC/BASIC results published in the review of that compiler in AC V2.9. The additional benchmarks were run on an Amiga with a 68010 processor, so the AC/BASIC results are about five to seven percent slower than the rest. Also note that the Sieve benchmark from that issue ran only one iteration, so the AC/BASIC result was multiplied by ten to get an approximation of its performance over ten iterations. The tests were timed by the programs themselves. The code for the F-Basic versions of these tests, minus the timing routine, is included to provide a glimpse of F-Basic's structure.

For the PSET test, the F-Basic program uses a high-level pixel drawing command, which apparently does some checking to make sure the parameters and window are appropriate. This slows the process a bit, but this program could have been written with the Amiga's WritePixel routine, which does no checking. This is how the Modula-2 program works, and the different designs account for most time disparity. "Single" and "Double" in the results refer to the floating point precision used. F-Basic offers only one precision, which, as the benchmarks show, is more precise than normal single precision and much faster.

F-Basic shows its youth through inconsistency. It has great string handling functions, but lousy strings. It has excellent event handling capabilities, but no SLEEP function. It has many valuable graphic routines, but no simple screen clearing procedure. It produces amazingly fast code which cannot be run from the Workbench and has its own albatross.

So is it worth it? If you need to produce professional quality programs, you will have to look elsewhere. F-Basic has too many flaws felt by the user—flaws that other compilers avoid. Someone paying for a program does not want to pay for the developer's choice of compiler as well. If you are looking for a complete, reasonably priced language for personal uses, however, F-Basic is a good choice. It lists for $79.95, which is much less than most compilers, and simplifies many otherwise difficult tasks.

Since this is an early version of the compiler, it is important to consider the support it will receive from its publishers. Two items in the F-Basic package helped form much of my opinion of Delphi Noetic. The first was two sheets of additions to the manual. The sheets were not the obligatory bug fixes, but explanations of several features added to F-Basic at the last minute. Apparently Delphi Noetic is not content to release a compiler, then sit back and count its cash. Constant revisions are almost a requirement for compilers, and Delphi Noetic shows every indication of providing this support.

The second item was a note near the end of the manual. For a fee, Delphi Noetic promises customized versions of its compiler. Send the developer specific requirements, and they return an estimated cost. This is a great offer for everyone and an observable indication of what Delphi Noetic thinks of its customers. Some features described in the additions to the manual were results of this policy.

If you buy F-Basic, you receive a product that needs improvement, but you also get an apparent commitment to undertake that improvement. I look forward to a more mature F-Basic, as well as additions to its environment. If its deficiencies are addressed, F-Basic could take a place among the most powerful Amiga languages.

F-Basic $79.95
Delphi Noetic Systems, Inc.
P.O. Box 7722
Rapid City, SD 57709
(605) 348-0791

•AC•

DISKMASTER

Point and Click Simplicity Comes to the File Management Arena

Reviewed by Steve Hull
GEnie: LightRaider
People Link: St.Ephen

I'll never forget my introduction to AmigaDOS. I was impressed by Intuition's point-and-click interface, but I knew that (despite what the salesman said) the Amiga's true power could only be tapped through the CLI.

Proudly, I flipped on my newly-assembled system and double-clicked on the CLI icon. After removing the Workbench disk from my one-drive system, I replaced it with a public domain software disk I had purchased that day. Though not familiar with AmigaDOS, I figured I'd seen enough disk operating systems to at least examine a disk listing. I typed D-I-R, and hit return. A requester popped up: "Please Insert Volume Workbench in any drive." Puzzled, I complied—only to get a directory listing of the newly-inserted Workbench disk!

Several "Unknown command" DIRs later, I realized AmigaDOS was a whole different creature than what I was used to. And a royal pain to use on my unexpanded 512K system.

The Amiga's first buyers were, by and large, technically proficient individuals who could not resist a programmer's dream machine that could be placed on a desktop. By nature, such people are difficult to intimidate. It's a different story, though, for the type of buyer on whom the Amiga's success ultimately hinges—the home user. These individuals are often attracted by the Amiga's bright icon interface which offers a friendly contrast to the MS-DOS world's cold "A>" prompt. While such users would benefit from learning their way around AmigaDOS, should it be mandatory?

DMaster's main screen features source and destination windows split by a strip of AmigaDOS-like function buttons.

Progressive Peripherals and Software's latest utility, DiskMaster, allows the techno-terrified to have their cake and eat it, too. With Diskmaster, even a rank beginner can list, rename, delete, and copy files through the comfortable point-and-click interface.

DiskMaster requires 512K of memory and one disk drive. It is not copy protected in any way, and the manual recommends that you make working copies and store the master disk in a safe place.

Clicking the DiskMaster icon from Workbench brings up the main DiskMaster screen, which is divided into three parts; a "source" window and a "destination" window, divided by a strip of function buttons, most of which approximate AmigaDOS functions.

(continued)

Using DiskMaster couldn't be simpler. To view the contents of any disk in the internal drive, just insert the disk and click once on the DF0: button. The contents of the disk—directories and files—are displayed. Directory names appear in a different color than filenames. To examine a directory, simply double-click its name.

Simple AmigaDOS functions such as copy, delete, and rename are similarly easy to use. Single-drive owners can especially benefit by using DiskMaster to transfer files quickly and easily via the RAM disk. Since DiskMaster loads into memory all at once, you need not leave the program disk in the drive once you have loaded. For a complete list of DiskMaster's functions, refer to the sidebar.

DiskMaster bristles with little convenience features that enhance its utility. A sans-serif "clean" font is built into the program—no special font need be installed. You can also select a special "small" font, allowing more characters per line, or you can shift into interlace to double the amount of lines displayed. Choosing "half-height" limits the DiskMaster display to the lower half of a hires screen. Devices supported default to DF0:, DF1:, and RAM:, but you can define up to twelve different devices. Once you have your copy of DiskMaster set up exactly as you like it, selecting SAVE CONFIG from the CONFIGURE menu ensures that the program initializes to your settings in future sessions.

Diskmaster satisfies the needs of a wide range of users. People running one-drive systems can appreciate having the power of AmigaDOS without constant, cumbersome disk swapping. People with large disk collections find that Diskmaster's responsive interface makes disk organization a breeze. Hard disk users can browse and search through megabytes of files in seconds on a screen that displays up to 70 filenames at once.

Ironically, DiskMaster's programmer, Greg Cunningham, may have worked himself out of some sales by programming the fine public domain and shareware variations that initially caught Progressive Peripherals' attention. Those who want Greg's best—in this case, a fully-featured Cadillac that makes other disk utilities look like Hyundais—definitely want to check this one out.

DISKMASTER FUNCTIONS

Dir
Print directory
Rename (Directories and files)
Copy (Directories and files)
Delete (Directories and files)
Update (Date/time stamp)
File comments
MakeDir
Disk copy
Disk Format
Run program
Protect (Archive/Read/Write/Execute/Delete)
Filename pattern search
Show IFF image
Play IFF sound
Read text files

Notes:
Up to 12 device names (i.e., DF0:, VD0:, HD0:) can be assigned to control buttons

Add, list, and extract archive functions are available through menu selections, but require an external archiving program (available in public domain/shareware) to run.

Up to six more external commands may be assigned to control buttons; such commands must be stored in the system disk's c: directory.

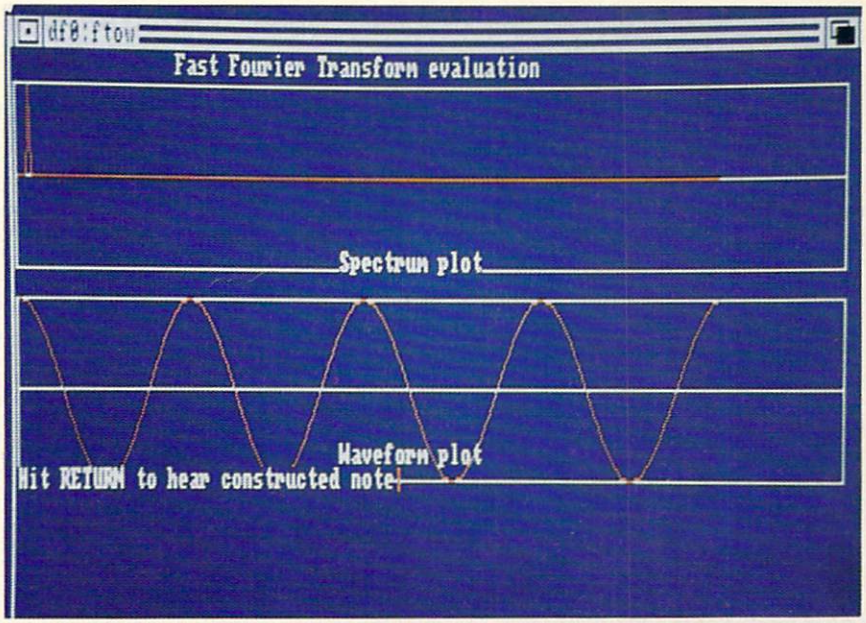
DISKMASTER
Progressive Peripherals & Software, Inc.
464 Kalamath Street
Denver, CO 80204
(303) 825-4144
Suggested price $49.95
Not copy protected

•AC•

Digital Signal Processing *in AmigaBASIC*

by Robert Ellis

The use of computers to examine and alter analog signals is known as digital signal processing. A small section of this field is a key part of today's electronic music. The Amiga, with its built-in sound generation system, surpasses other personal computers in this respect. Today, there are a large number of programs on the market which allow the Amiga owner to use his computer to generate very sophisticated music and sound patterns. These programs allow the user to create and alter tones by editing frequency and time relationships in the waveforms sent to the tone generation system. This article presents a behind-the-scenes look at how these transformations from numbers to sounds are accomplished. It also will allow the reader to perform his own digital signal processing experiments with the help of Fast Fourier Transforms (FFT).



Sounds, tones, and music are made up of periodic or repeating waves. These periodic waves, as they exist in the physical world, are continuous-time in nature. Computer generated tones, on the other hand, are discrete-time in nature. The term discrete-time means that the tones, or waveform data are generated from a set of discrete numerical values held in computer memory. These discrete values are presented to a digital-to-analog converter at a set rate. The stepwise construction of the signal is passed to an audio system such as an amplifier and a set of speakers. The human ear picks up this stepwise approximation, and along with the brain, translates it back to what is perceived as a continuous-time signal.

To generate the sound of a musical note from AmigaBASIC, a waveform must be constructed in an integer array. This array is 256 entries long and contains values for each entry, which are no less than -127 and no greater then 128. The waveform is played repeatedly by the sound generation circuitry to produce a note. This data array which makes up the waveform is a perfect example of a discrete-time signal. The waveform is made up of individual integer values (or discrete elements). These elements are known as a samples. The samples have a limited range of values, set by the hardware. These discrete-time signals are often illustrated graphically. You should be able to fill in the lines to form a sine wave from this picture. The dot at the end of each line represents the discrete value for that sample point, which is held in the integer array.

When repeated continuously the sample will have a frequency associated with it. The note at A above middle C has a frequency of 440 cycles per second (cps). The frequency is also known as a frequency spectrum value, or spectral point. If a waveform is comprised of one sine wave at a given frequency, it has only one spectral point. A waveform comprised of two sine waves added together will have two spectral points. Waveforms, which are not pure sine waves (like square waves) will have a set of spectral points.

(continued)

There are many ways to construct a waveform in an array in BASIC. One method uses a FOR NEXT loop to set the amplitude of each of the 256 elements. This waveform could then be used in conjunction with the WAVE and SOUND statements in BASIC to produce a tone. Or one might copy the contents of a small file into the data array limiting the values to the proper range. This last method will produce strange noises in most cases.

Another way to produce waveforms is to translate frequency domain information into time domain information. The mathematical translation from one domain to the other is accomplished by using Fourier transforms. Fourier transforms were discovered by clever Frenchman J. B. J. Fourier.

Fourier discovered that any periodic function could be described by an infinite series of sinusiods of harmonically related frequencies. Fourier's analysis is also known as frequency analysis. The Fourier integral expresses the summation of frequency values, phase angles, and frequency magnitudes.

$$H(f) = \int_{-\infty}^{+\infty} h(t)\ e^{-j\,2\,\pi\,f\,t}$$

In the equation above, H(f) is the representation of the signal in the frequency domain and h(t) is the representation of the signal in the time domain. This equation can be expanded to accommodate a computer algorithm. The following substitution is needed for this expansion:

$$e^{-j\,O} = COS(O)\ \ -j\ SIN(O)$$

This expands the original integral to:

$$H(f) = \int_{-\infty}^{+\infty} h(t)\ \ (\cos(2\,\pi\,f\,t) - j\,\sin(2\,\pi\,f\,t))$$

Finally, the integral is expressed as a summation. This summation is known as the Fourier Series. The equation below shows a periodic waveform is comprised of a set of harmonically related frequencies with finite amplitudes.

$$H(f) = \frac{a0}{2} + \sum_{n=1}^{n->infinity} [\ a\ n\ \cos(2\,\pi\,n\,f0\,t) + b\ n\ \sin(2\,\pi\,f0\,t)]$$

where f0 = 1/T0 which is the fundamental frequency of the periodic waveform.

These equations simply state that if you know the set of frequencies and their amplitudes that make up the waveform, you can reconstruct that waveform. This process can be thought of as waveform synthesis. This also means if you reverse this process you can find the frequencies and the amplitudes that make up a periodic waveform. This is also known as spectrum analysis.

The equations listed above have been translated into a computer algorithm known as a Fast Fourier Transform. This algorithm is responsible for turning waveforms into frequency spectrum and frequency spectrum into waveforms. The basic implementation of this algorithm is named FFT1. This routine has a set of requirements the calling program must adhere to. Those requirements are as follows:

1) The number of samples on which the algorithm will operate must be a power of 2 (i.e. 64, 128, 256...).

2) The array containing the input data, whether the waveform samples or the frequency spectrum samples, must be twice the size of the number of samples.

3) The input data array must be comprised of DOUBLES size elements.

The FFT algorithm has been implemented in BASIC to allow the user the simplest access to the Amiga's tone generation hardware. However, implementation in Basic causes the FFT operation to be quite slow in calculation (about 30 seconds for a 256-point FFT). This article provides the listing of the two example programs and the listing for the FFT implemented in C. With SIN and COS tables, this algorithm can also be implemented in assembler. The algorithm is listed below.

FFT1 Algorithm:
[STEP 1] *Initialize variables used in the algorithm.*
[STEP 2] *Perform a pre-weave butterfly operation of the input data array.*
[STEP 3] *Update loop counters and index variables, check for completion; IF YES fall through to STEP4, ELSE branch back to STEP 2.*
[STEP 4] *Set up multiplication coefficients.*
[STEP 5] *Perform weave (butterfly) operation and multiply each weave operator by coefficients.*
[STEP 6] *Check for completion. IF NOT complete increment loop counters, indexes, form new multiplier coefficients, and branch back to STEP 5, ELSE fall through to STEP 7.*
[STEP 7] *Control falls through to this point when secondary weave is complete, and control is returned to the caller of FFT1.*

Note the four main sections of this algorithm are preceded by a comment block identifying each section. The most complex parts of the algorithm are the weaving sections. The first weave intermixes the elements of the array, keeping the real and imaginary parts of the complex number separate. The second weave causes the translation between the time and frequency domain, by intermixing real and imaginary parts of each complex element of the array multiplied by a changing coefficient.

Now it is time to put the FFT to work. The first task for the FFT is to generate the correct waveform data for a set of frequency spectrum entries. To do this correctly, first insure the correct input data are presented to the FFT subroutine. These steps are as follows:

1) The frequency indexes chosen are integer multiples of the first frequency entered fundamental.

2) All entries fall in the range from +1.0 to -1.0.

3) All entries should be made to even-numbered indexes in the data entry array. This constitutes the real portion of a complex number array. The odd-numbered index entries constitute the imaginary portion of the complex number array.

The programming steps listed below have been implemented in the FTOW base program (Listing 1). These steps will allow the user of the FFT1 algorithm to correctly generate new sound waveforms. In this program, the user inputs the spectral frequencies and the amplitude values directly into the data array at the start of the program.

1) Dimension a DOUBLE array of 512 bytes.

2) Zero out the array.

3) On an even index (real portion of the complex number), enter the frequency spectrum points into the array. The values of these data entries should be between -1.0 and +1.0. (Index for read data points start at a value of 2.)

4) Call the FFT routine.

5) Scan the even elements of the result array and form a scaling factor for a maximum value of 128 and minimum value of -127.

6) Transfer the waveform data from the even bytes of the DOUBLE array to a 256 byte array of integers, multiplying by the scaling factor.

7) Call the built-in WAVE function in BASIC with the INTEGER array as its input.

8) Call the built-in SOUND function in BASIC to hear the constructed waveform.

When the program is run, the screen will be cleared and two display boxes will be drawn. The top display box will contain a spectrum plot of the input data. This will show the user the frequency values and the relative amplitudes he has chosen. You should note that if you enter a spectral component with a negative amplitude, the resulting waveform will differ from that

(continued)

when the spectral point had a positive amplitude. This is because the frequency component is being added into the waveform calculations out of phase with the other entries. You should experiment with different amplitudes, but keep the frequency values the same.

The second display box will contain the constructed waveform which will be scaled before displaying, so the largest amplitude in the waveform will be a full scale reading in the display box.

The next step in the program allows the user to listen to the constructed waveform. To do this, transfer the complex data array values to the integer array to be passed to the WAVE subroutine. Before transferring the real portion of the data to the packed integer array, scan the array for the maximum value. This maximum value will be used as a scaling factor during the data transfer, to prevent the data from falling out of the range of the tone generation system limits (-127 to 128).

The waveform information is transferred to channel number 2 in the Amiga hardware. It is placed in channel 2 so the default tone in channel 0 is preserved and both tones are directed to the same channel. For comparative purposes, the program will play the default tone (which is a single frequency sine wave) and then play the newly constructed tone.

Once you understand the function of the program, you can add file utilities to the program so that constructed tones can be saved. Loading waveforms from a file for use in this program or in another program will be much faster than generating them through the FFT algorithm each time.

Some example data points have been selected with known good results. These examples are in the top of the FTOW program. They have been commented out so that they do not interact with each other at all. Try each of the examples by deleting the comment in the first character of each line and run the program. Each time you move on to the next example, remember to comment out the current example data statements so they will not interact.

The second program WTOF will use the FFT algorithm to translate time domain information to frequency domain information. This function is known as spectrum analysis. There are many instruments in the engineering market today that will perform this function in real time. These instruments are manufactured by companies such as HP, Techtronics, IFR, and Eaton. Prices range from $10,000 to $60,000. Although the AMIGA may not be as fast as or have all of the features of these dedicated devices, it will allow you to experiment in this area of electrical engineering.

As with the FTOW program, you will find a section of example data at the beginning of the main section of the program. To examine the results of the translation from the time domain to the frequency domain, you must uncomment the data entry statements and run the program.

When the program executes, the top display, as before, will show the input data graphically. The program will then calculate the frequency spectrum from the time domain information. The frequency spectrum will be displayed in the second graphics window.

One difference between the output of this program and the first program is the effect of aliasing. This is where frequency information displayed will seem to have a mirror image at about the halfway mark. This effect is present in the frequency-to-time conversion also, but is not seen because the waveform generated is symmetrical and periodic. Another name for this effect is the Nyquist frequency.

This article has been an introduction to Fourier Transforms and the conversion between the time and frequency domains. This transformation has been accomplished by the use of an FFT subroutine which was implemented in Amiga Basic and C. Although the implementation is not fast enough for real time data analysis, the same methods apply. To form a real time spectrum analyzer, the FFT routine and data zeroing routine should be implemented in assembler with care taken to make sure they execute as fast as possible.

For those interested in a more in-depth look at digital signal processing, or just Fourier Transforms, there are numerous books on the subject. A few of these books are listed in the references. In addition to the books listed, are other books available on the subject of digital filters will include at least one chapter on Fast Fourier Transforms.

References:

Network Analysis, M.E. Van Valkenburg
published by Prentice-Hall

Digital Signal Processing, Alan V. Oppenheim, Ronald W. Schafer
published by Prentice-Hall

Reference Data For Radio Engineers, ITT
published by Howard W. Sams

Musical Engineer's Handbook, Bernie Hutchins
published by Electronotes

Listing One

```
' ************************************
'*                                    *
'* File:      WTOF                    *
'* Function:  Demonstrate use of Fast *
'*            Fourier Transforms to    *
'*            to generate frequency    *
'*            spectra from waveform    *
'*            input data.              *
'* Date:      March 1988              *
'* Author:    Robert Wm. Ellis        *
'*                                    *
' ************************************
```

```
'*****
' Global variables
'*****
DEFDBL a,b,c,d,w,t
DEFINT i,j,k,n,s,o
DIM  iwaveform%(255)
DIM a(1030)


'*****
' Main program entry point
'*****

n = 256          ' Number of points in FFT
size = 128

CALL init        ' Set up initial screen

'
' Clear the data array before entering any
' values into the REAL (even indexes) portion
' of the array.
FOR i=0 TO n*2
  a(i)=0
NEXT i

' EXAMPLE # 1 - SQUARE WAVE
' This will generate a square wave as input,
' producing odd harmonics in the frequency
' spectrum.  The waveform is entered to be
' symetrical around zero TIME as if it was
' a COS function.
'
rval = 1
FOR i=0 TO n*2 STEP 2
  IF( (i+32) MOD 64) = 0 THEN
    rval = rval * -1
  END IF
    a(i) = rval
NEXT i


' EXAMPLE # 2 - IMPULSE FUNCTION
' This will result in an example of an infinitely
' thin pulse in the time domain producing an infinite
' frequency response.  The output in the frequency
' domain will be a line across the top of the Output
' Spectral display.
'a(2) = 1

' EXAMPLE # 3 - FINITE IMPULSE FUNCTION
' This will result in a frequency spectrum which
' has some amount of energy at almost every frequency
' range.  The phase of the frequency information
' will change from IN PHASE to OUT of PHASE in an
' oscillating pattern.
'FOR i=2 TO 16 STEP 2
'  a(i) = 1
'NEXT i


' EXAMPLE # 4 - SINE WAVE
' This example will a sine wave at a single frequency,
' which will produce a single spectral output.
' By changing the value of sl you can see that
' a higher frequency will produce a spectral point
' that is farther away from the beginning of the
' the spectrum (D.C.).
'sl = 16
'wseg = 6.28319 / sl
'j = 1
'FOR i=0 TO n*2 STEP 2
'  wtemp = ((i/2) MOD sl) * wseg
'  a(i) = COS(wtemp) * .1
'NEXT i

'*********
' Display the input waveform, call for the
' transformation between the time domain and
' the frequency domain, and display the spectrum
```

(continued)

```
' plot of the resulting frequencies.

CALL plotu(a(),n,40)
CALL fft1(a(),n,1)
CALL plotu(a(),n,110)

END

'**** end of main program ****

'*****************
' Subroutine  : init
' Function    : Setup up program
' Inputs      :
'*****************
SUB init STATIC

CLS

PRINT PTAB(120);"Fast Fourier Transform evaluation"
LINE(0,10)-(600,70),,b
LINE(0,40)-(600,40)
LOCATE 9,1
PRINT PTAB(220);"Input Waveform Data"
LINE(0,80)-(600,140),,b
LINE(0,110)-(600,110)
LOCATE  17,1
PRINT PTAB(220);"Output Spectral plot"
END SUB


'*****************
' Subroutine  : FFT1
' Function    : Perform Fast Fourier Transform
' Inputs      : darray - data array in time or frequency
'             : nn     - number of data points
'             : isign  - switch from forward or reverse trans-
form
'*****************
' local integer variables
'     ii,jj,n,mmax,m,j,i,istep
'
' local double variables
'     wtemp,wr,wpr,wpi,wi,theta,tempr,tempi
'
SUB fft1( darray(1), nn%, isign% ) STATIC


  n = 2 * nn
  j = 1

'*******
' Pre-weave butterfly operation is performed
' to incoming complex array.
'*******

  FOR i=1 TO n STEP 2
    IF( j > i) THEN
      tempr = darray(j)
      tempi = darray(j+1)
      darray(j)=darray(i)
      darray(j+1)=darray(i+1)
      darray(i)=tempr
      darray(i+1)=tempi
    END IF
    m=n/2
    WHILE( (m >=2) AND (j>m))
      j = j-m
      m = m/2
    WEND
    j = j+m
  NEXT i

  '
  ' Top of secondary weave operation
  '
  mmax = 2
  WHILE( n > mmax )

    '*******
    ' Calculate fixed wieghting coefficients
    '*******

    istep = 2 * mmax
    theta =6.28318530717959#/(isign*mmax)
    wpr = -2!*(SIN(.5*theta)*SIN(.5*theta))
    wpi = SIN(theta)
    wr=1!
    wi=0!

    '********
    ' Secondary weave operation is performed
    ' on complex array, and results are multiplied
    ' by coefficients.
    '********
    FOR m=1 TO mmax  STEP 2
      FOR i=m TO n STEP istep
        j=i+mmax
        tempr=wr*darray(j)-wi*darray(j+1)
        tempi=wr*darray(j+1)+wi*darray(j)
        darray(j)=darray(i)-tempr
        darray(j+1)=darray(i+1)-tempi
        darray(i)=darray(i)+tempr
        darray(i+1)=darray(i+1)+tempi
      NEXT i

      '********
      ' recalculate the coefficients
      '********
      wtemp=wr
      wr=wr*wpr-wi*wpi+wr
      wi=wi*wpr+wtemp*wpi+wi
    NEXT m
    mmax=istep
  WEND

END SUB


'*****************
' Subroutine  : wplot
' Function    : plot time domain data in waveform window
' Inputs      :
'*****************

SUB plotu(a(1),n%,offset%) STATIC

' Find maximum value in data array for scaling
' factor to be applied to each array element.

max = 0
FOR i=2 TO n*2 STEP 2
  IF ABS(a(i))> max THEN
    max = ABS(a(i))
  END IF
NEXT i
ascale = -30/max

LINE (1,a(2)*ascale+offset%)-(1,a(2)*ascale+offset%),3
FOR i=2 TO (n)*2  STEP 2
  LINE-(i,a(i)*ascale+offset%),3
NEXT i

END SUB  'end of subroutine wplot
```

Listing Two

```
'************************************
'*                                  *
'* File:      FFT                   *
'* Function:  Demonstrate use of Fast  *
'*            Fourier Transforms    *
'* Date:      March 1988            *
'* Author:    Robert Wm. Ellis      *
'*                                   *
```

```
' ************************************

'*****
' Global variables
'*****
DEFDBL a,b,c,d,w,t
DEFINT i,j,k,n,s
DIM  iwaveform%(255)
DIM a(520)


'*****
' Main program entry point
'*****

n = 256  ' number of points in FFT

CALL init

' clear spectrum array
FOR i=0 TO n*2
  a(i)=0
NEXT i

' EXAMPLE # 1 - Cosine wave
'   This example will produce a cosine wave with
' a fundimental frequency of 4.  This means that
' there will be 4 full waveforms in the output
' array.  Since the array is contructed of complex
' numbers (real and imaginary part), the entry is
' made at location 8 (+2).  The imaginary parts are
' located in the odd indexes in the array.
'
a(10)=1

' EXAMPLE # 2 - Cosine wave
'   This version of the cosine wave will be out
' of phase.  That means that it will start in the
' negative portion of the output.
'
'a(10)=-1

' Example # 3 - Sine wave
'   Here the imaginary portions of the frequency
' array is used to produce a sin function.  A small
' real value is needed so that a divide by zero
' error does not occur.
'
'a(10)=.001
'a(9)=-1


' EXAMPLE # 4 - Square wave
'   The frequency array is filled starting at
' element 2.  The fundimental frequency for this
' square way will be 8, so index 10 is the first
' entry point (8 + 2).  Since the square WAVE is
' comprised of ODD harmaonics, the next entry will
' be at index (8 * 3) + 2.  Since only 4 harmonics
' are used to synthesize the square wave there will
' be some ripple in the waveform.
'
' The following equation is the fourier series for
' the square wave:
'
' y(t)=2A[1/2 + (2/pi)cos(f0) - (2/3pi)cos(f3) +...]
'
'a(10)=.67
'a(26)=-.22
'a(42)=.12
'a(58)=-.08

' EXAMPLE # 5 - Triangle wave
'   To construct a triangle waveform from the fourier
' series the same steps are taken as in the square
' wave.  the coefficients for the triangle wave can
```

(continued)

```
' be determined by the following equation.
'
'                      2                  2
' y(t)=2A[1/2 + (2/pi) cos(f0) + (2/3pi) cos(f3) +...]
'
'a(10)=.4
'a(26)=.045
'a(42)= .0162


CALL plotu(a(),n,40)
CALL fourier(a(),n,-1)
CALL plotu(a(),n,110)

CALL play(a(),iwaveform())

END


'*****************
' Subroutine  : init
' Function    : Setup up program
' Inputs      :
'*****************
SUB init STATIC

CLS

PRINT PTAB(120);"Fast Fourier Transform evlauttion"
LINE(0,10)-(600,70),,b
LINE(0,40)-(600,40)
LOCATE 9,1
PRINT PTAB(240);"Spectrum plot"
LINE(0,80)-(600,140),,b
LINE(0,110)-(600,110)
LOCATE  17,1
PRINT PTAB(240);"Waveform plot"
END SUB



'*****************
' Subroutine  : fourier
' Function    : Perform Fast Fourier Transform
' Inputs      : darray - data array in time or frequency
'             : nn     - number of data points
'             : isign  - switch from forward or reverse trans-
form
'*****************
' local integer variables
'    ii,jj,n,mmax,m,j,i,istep
'
' local double variables
'   wtemp,wr,wpr,wpi,wi,theta,tempr,tempi
'
SUB fourier( darray(1), nn%, isign% ) STATIC


  n = 2 * nn
  j = 1
  FOR i=1 TO n STEP 2
    IF( j > i) THEN
      tempr = darray(j)
      tempi = darray(j+1)
      darray(j)=darray(i)
      darray(j+1)=darray(i+1)
     darray(i)=tempr
     darray(i+1)=tempi
    END IF
    m=n/2
    WHILE( (m >=2) AND (j>m))
      j = j-m
      m = m/2
    WEND
    j = j+m
  NEXT i

  mmax = 2
  WHILE( n > mmax )
    istep = 2 * mmax
   theta =6.28318530717959#/(isign*mmax)
   wpr = -2!*(SIN(.5*theta)*SIN(.5*theta))
    wpi = SIN(theta)
    wr=1!
    wi=0!

   FOR m=1 TO mmax  STEP 2
     FOR i=m TO n STEP istep
       j=i+mmax
      tempr=wr*darray(j)-wi*darray(j+1)
      tempi=wr*darray(j+1)+wi*darray(j)
      darray(j)=darray(i)-tempr
      darray(j+1)=darray(i+1)-tempi
      darray(i)=darray(i)+tempr
      darray(i+1)=darray(i+1)+tempi
     NEXT i
     wtemp=wr
     wr=wr*wpr-wi*wpi+wr
     wi=wi*wpr+wtemp*wpi+wi
   NEXT m
   mmax=istep
 WEND

END SUB



'*****************
' Subroutine  : play
' Function    : play waveform constructed by FFT
' Inputs      :
'*****************
SUB play(a(1),waveform%(1)) STATIC

max = 0
FOR i=0 TO 512 STEP 2
  IF ABS(a(i))> max THEN
    max = ABS(a(i))
  END IF
NEXT i

scale = 127/max

FOR i =0 TO 255
  waveform%(i)= a((i+2)*2) * scale -1
NEXT i

SOUND 440,70
INPUT "Hit RETURN to here constructed note",b
WAVE 2,waveform%
SOUND 110,70,127,2


END SUB

'*****************
' Subroutine  : plotu
' Function    : plot time domain data in waveform window
' Inputs      :
'*****************

SUB plotu(a(1),n%,offset%) STATIC
max = 0
FOR i=2 TO n*2 STEP 2
  IF ABS(a(i))> max THEN
    max = ABS(a(i))
  END IF
NEXT i
scale = -30/max

LINE (1,a(2)*scale+offset%)-(1,a(2)*scale+offset%),3
FOR i=2 TO n*2  STEP 2
  LINE-(i,a(i)*scale+offset%),3

NEXT i
END SUB  'end of subroutine wplot
```

Listing Two

```
/*************
 *  Procedure :  FFT1
 *  Function  :  Perform Fast Fourier Transform
 *  Inputs    :  data  - data array of complex double
 *            :  nn    - number of valid data points
 *            :isign - switch forward to reverse transform
 *************/

fft1( double data[], int nn, int isign )

{
static int       ii,jj,n,mmax,m,j,istep,i;
static double    wtemp,wr,wpr,wpi,wi,theta;
static double    tempr,tempi;

  n = 2*nn;
  j = 1;
  for( ii = 1; ii <= nn; ii++)
  {
     i = 2*ii - 1;
     if( j > i )
     {
        tempr = data[j];
        tempi = data[j+1];
        data[j] = data[i];
        data[j+1] = data[i+1];
        data[i] = tempr;
        data[i+1] = tempi;
     } /* end of if */

     m = n / 2;
     while( (m >= 2) && (j > m))
     {
        j = j - m;
        m = m / 2;
     }  /* end of while */
     j = j + m;
  } /* end of for */


  mmax = 2;
  while( n > mmax )
  {
     istep = 2 * mmax;
     theta = 6.283185/(isign * mmax);
     wpr = -2.0*( sin(0.5 * theta ) * sin(0.5 * theta));
     wpi = sin(theta);
     wr = 1.0;
     wi = 0.0;


     for( ii = 1; ii <= (mmax / 2); ii++)
     {
        m = 2 * ii - 1;
        for( jj = 0; jj <= ((n-m) / istep); jj++ )
        {
           i = m+ jj * istep;
           j = i + mmax;
           tempr = wr * data[j] - wi * data[j+1];
           tempi = wr * data[j+1] + wi * data[j];
           data[j] = data[i] - tempr;
           data[j+1] = data[i+1] - tempi;
           data[i] = data[i] + tempr;
           data[i+1] = data[i+1] + tempi;
        }/* end of for */
        wtemp = wr;
        wr = wr * wpr - wi * wpi + wr;
        wi = wi * wpr + wtemp * wpi + wi;
     } /* end of for */
     mmax = istep;
  } /* end of while */
} /* end of procedure fft1 */
```

•AC•

Produced in association with
Commodore Business Machines

Commodore Business Machines will be there – with a giant display of exciting computer hardware and software. Hundreds of other manufacturers, distributors and retailers will also be there – showing and selling the entire spectrum of Commodore and compatible products.

Top experts in home and business computing will be there – leading seminars, workshops and demonstrations which are free with admission for all show visitors.

The Amiga, the C-64, the C-128, the Commodore PC line, all the major peripherals, programs, accessories – many more to be introduced – they're all at The World of Commodore in Philadelphia. All in one place for four days.

This is the show everyone's been waiting for. The first-ever Commodore-sponsored computer extravaganza in the United States.

Don't miss this once-a-year opportunity to shop for incredible bargains as leading computer companies clear out millions of dollars worth of inventory.

Discover innovative products and services for your business, for home, for school, for your creative life.

Get in on the excitement.

Adults $10 Students & Seniors $8

Admission price includes seminars and stage presentations.

Thurs. Nov. 3 & Fri. Nov. 4 10 am – 9 pm
Sat. Nov. 5 & Sun. Nov. 6 10 am – 6 pm

Exhibitors, contact:
The Hunter Group
(416) 595-5906
Fax (416) 595-5093

The Canadian World Of Commodore is in its sixth successful year at the Toronto International Centre, December 1 – 4, 1988.

The Developing AMIGA

by Stephen R. Pietrowicz

Writing demonstrations of your software

You've just finished writing a program you're convinced will blow the doors off the Amiga market. How do you get the word out without spending a lot of money?

Ideally, you'd like people to be able to try your program so they'll have an idea what it's like before they buy it. Unfortunately, not all computer stores will demo software packages for customers. Your program just might sit up on the shelf waiting for someone to buy it.

Some companies have decided the best way to show potential customers a new product is by demonstrating the software. This demo may be a slideshow of IFF pictures, an animations, or crippled copy of the actual program.

A variety of different programs can help you write an extremely effective demo. ShoWiz, a public domain program by J.L. White, is a slideshow program that will display every IFF screen in a directory. ShoWiz displays pictures by fading in and out, wiping, checkerboarding, and over 30 other methods. ShoWiz reads display instructions from a script file and displays pictures randomly.

The Director, by Keith Doyle of the Right Answers Group, is one of the better programs for writing demos. This program was used to write "RGB Hazard", the first place winner in the Badge Killer Demo Contest. I recently used Director to write a demo of ACO (The Amiga Conference) which ran at the PeopleLink booth during AmiExpo in Chicago last July. You combine sound and animation to scripts very easily using Director's BASIC-like script language.

Projector, a player program used to run the demos the Director produces,is freely distributable so you can give anyone copies of your work. If you think about it, programs like Projector really enhance the products they support. They allow people to see what a product can do, and at the same time can also entice them to buy the product. Programs like SONIX, the animations from Hash Enterprises, Videoscape-3d, and others have players associated with them.

If you distribute a demo of your product, the demonstration should be as close to the actual product as possible,without actually giving the user the full functionality of the software. What does that mean? It means if you're writing a crippled version of your desktop publishing software, let the user do everything the real program does except print. Now, granted, disabling critical parts of the program is just common sense, but you'd be surprised...

For example, I saw a demo for a paint program distributed by the company that wrote it. Nearly all the features were enabled, except the "SAVE IFF SCREEN" function. They really shouldn't have distributed the program as a player. Programs like GRABBiT and HERMiT, which can capture an IFF screen and write it to a file, totally defeat the "crippled" part of the demo!

Another paint program demo I saw recently disables more features, but still lets the user understand how the program works. The screens can still be captured, but consumers may be more inclined to buy it since not all of the features are enabled in the program.

Remember: Demonstrations are meant to entice customers into buying your product and can be a very valuable asset.

And Now for Something Completely Different....Usenet

Usenet is the loosely connected network of computers that exchanges electronic mail and news throughout the world. Literally hundreds of machines and thousands of people read the various newsgroups carried by the Usenet news system.

When a user posts a notice to the Usenet, the news system on the host machine transmits the message to each of the other sites to which it is connected. News is transmitted from machine to machine through the entire network. Estimates put the amount of data transmitted to systems via Usenet at 2.5 - 3 megabytes per day!

Newsgroups are split into several different categories:

comp - Computer related topics: computer systems, languages, operating systems, etc.

misc - Miscellaneous topics: consumer issues, sale notices, job related info, etc.

news - Usenet news information: Usenet administration tips, new Usenet site announcements, etc.

rec - Recreational topics: photography, games, ham radio, scuba diving, etc.

sci - Science related topics: physics, space, math, medical, etc.

soc - Social issues: college, men, women, singles, etc.

talk - Discussions on controversial topics: abortion, politics, religion, etc.

Each categories is broken down into sub categories, depending upon the topic. For example, the computer graphics subgroup is called comp.graphics. If the topics warrant further specialization, they simply extend the newsgroup name. (ie., comp.lang.c is the newsgroup for the C programming language).

Way back in 1985, before the current naming scheme was in place, groups were all prefixed with "net". One day, a new newsgroup started. It was called "net.micro.amiga". One of the first people (if not THE first) to post to this group was Eric Lavitsky, now with ASDG. Eric wrote many notices to net.micro.amiga, telling everyone about the fantastic new machine he had just purchased. Slowly, as more people started finding out about the Amiga, an increasing number of notices were posted. Today, Amiga newsgroups are among the most popular on the Usenet.

Four Usenet newsgroups carry Amiga information:
comp.sys.amiga - General Amiga related topics
comp.sys.amiga.tech - Technical Amiga hardware/software topics
comp.sources.amiga - Source code for Amiga programs
comp.binaries.amiga - Binary versions of Amiga program in comp.sources amiga

If you'd like to start reading the Amiga newsgroups on Usenet, check with your employer or college. Many companies and colleges are sites on the Usenet and have computer accounts that log-in to access the news. Check with your site administrators for more information.

If you don't have news, check for Usenet sites in your area. Many sites on Usenet will let your site receive news from them upon request. You'll need the appropriate hardware and software. Just about any UNIX system can run the news software and the UUCP protocol (Unix to Unix CoPy). UUCP transmits news to different computer systems. If your machine supports this protocol, you can receive and send data from the Usenet.

Here are a few public access systems you can dial into to read the Usenet newsgroups (see table below).

Guess what? The news software and UUCP is being ported to the Amiga! The software is currently being beta tested, and should be released by press time. You'll probably see quite a few more Amigas used as Usenet sites in the future! As soon as the software is released, I'll let you know how you can get it.

Wrap Up

What type of information would you like to see in this column? I'd like some feedback from you! If you have any news, questions, or programming hints, send them in. The address is:

The Developing Amiga
c/o Amazing Computing
P.O. Box 869
Fall River, MA 02722

I can also be reached via People/Link. Send e-mail to CBM*STEVE.

•AC•

Stephen R. Pietrowicz is an assistant chairman of People/Link's Amiga Zone, author of ACO—The Sound and Graphics Conferencing Program, a freelance writer, and a member of The C Group.

System name	City, State	Phone #	Baud rates
pnet02	Redondo Beach, CA	213-376-5714	3/12/24
igloo	Northbrook, IL	312-272-5912	3/12/24
pern	New Orleans, LA	504-466-9109	3/12/24
pnet51	Minneapolis, MN	612-929-6699	3/12/24
magpie	New York City, NY	212-420-0527	3/12/24/96
bucket	Portland, OR	503-254-0458	3/12/24
pebco	Philadelphia, PA	215-956-0470	12
killer	Dallas, TX	214-824-7881	3/12/24
sugar	Houston, TX	713-438-5018	3/12/24

C Notes From the C Group

by Stephen Kemp, PLINK ID: SKEMP

You may often find that programs require you to perform some task repetitively (or they can at least be optimized by repeating such tasks). C provides several different "loop" type instructions for such occasions. This time we will discuss each kind of loop, along with several other statements important to loops. Before we discuss the individual statements, however, let me offer a refresher on compound or block statements.

All loop instructions, such as the "if" statement, assume that the next sequential instruction (or substatement) is under the loop instruction's control. Often, the controlling loop statement must handle a series of substatements—this is when you need the block format. You may remember that braces { } indicate statements contained in a block. Included after the initial looping statement, the open brace indicates that all instructions occurring before the closing brace are substatements of the first instruction. This explanation may seem a little vague now, but it should become clear when we get to some examples.

The first, and probably most often used, looping instruction we will discuss is the "for" statement. C programmers usually refer to this statement and its associated substatements as a "for-loop." We used a for-loop last month, but it won't hurt to discuss it again. The syntax of the "for" statement is:

```
for ( expression 1; expression 2; expression 3)
        substatement;
```

Expression 1 is normally used for loop initialization. This statement is executed before any other expression in the loop, so it is perfect for setting up the for-loop's initial parameters. For instance, if the loop is going to count, we might make a statement like "var = 1;" the first expression. (An expression 1 in the for-loop is optional.) The semicolon indicates the division between the expression 1 and expression 2 area, even if you do not include the first expression.

You can actually include several statements as the initialization expression. You do this by separating the statements with commas, rather than semicolons (as you would with normal statements). Remember, the semicolon after expression 1 indicates where expression 2 begins. These examples demonstrate the various ways to use the expression 1 area.

```
for(var = 1; expression 2; expression 3)
        substatement;

for( ; expression 2; expression 3)
        substatement;

for(x = 1,y = 1,z = 0; expression 2; expression 3)
        substatement;
```

The second expression is evaluated as true or false before each iteration of the loop. Remember, false is zero, and true is any non-zero value. If expression 2 is true, the loop's substatement is executed. When expression 2 evaluates false, the for-loop is terminated, and execution begins with the next instruction beyond the for-loop controls. Expression 2 is evaluated before the first iteration of the loop. Therefore, if expression 2 is false after evaluating expression 1, the loop is not executed.

Using the counting example again, the second expression might be something like "var < 100;". As long as the variable is less than 100, the loop continues. Expression 2, like expression 1, is optional, but the semicolon must be included to indicate the separation of expression 2 from expression 3. If the second expression is omitted, the for-loop is taken to be permanently true (commonly called a "forever" loop). You may think that this situation should be avoided, but forever loops can actually be very useful. Later, we'll look at several other ways to terminate a loop without depending on expression 2. Here are a couple of examples of how to use expression 2 in for-loops.

```
for(x = 1; x < 100; expression 3)
        substatement;

for( ; ; expression 3)
        substatement;
```

It's fairly obvious what task the third expression usually performs. Since expression 1 normally initializes the variables

used in expression 2, and expression 2 evaluates whether to enter the loop, expression 3 must change the parameters evaluated in expression 2. This instruction is executed after each iteration through the loop.

With a counting for-loop, a statement like "var++" increments the variable each time before evaluating expression 2 again. As with the other two expressions, this one is also optional. Notice though, expression 3 is not followed by a semicolon because it is the last expression of the "for" statement. Similar to the first expression, multiple statements can be included here, separated by commas (NOT by semicolons). Take a look at a few examples of how expression 3 fits into the for-loop instruction.

```
for( x = 1; x < 100; x++)
        substatement;

for( ; ; )
        substatement;

for(x=1,y=1,z=1; ; x++, y++, z++)
        substatement;
```

The substatement following a for-loop instruction is executed each time expression 2 is true. Unless you use the compound statement feature discussed at the start of this column, only one statement is executed during each iteration. When more statements are necessary, the open and close braces can be used to enclose all statements. The open brace must appear between the closing parenthesis of the for-loop instruction and before the first statement. The closing brace is placed anywhere after the last statement the for-loop should control. Although no specific formatting syntax is required when coding for-loops, I recommend you use one of the following methods so the area the for-loop controls is easily identified.

```
for(expression 1; expression 2; expression 3) {
        substatement;
        substatement;
        substatement;
}

for(expression 1; expression 2; expression 3)
{
        substatement;
        substatement;
        substatement;
}
```

To repeat (We are talking about loops!), a for-loop normally has 1 or more statements executing until the second expression is evaluated false, or the loop is terminated through some other means. Occasionally you do not need any statements other than one or more of the expressions in the syntax definition. To tell the compiler that no statements follow the for-loop, include a semicolon after the "for" instruction's closing parenthesis. This technique is demonstrated in the following example.

```
for(x = 0; var(x) != 0; x++) ;
```

This for-loop counts how many values an array has before a value of 0 is encountered. Since the expressions of the "for" definition handle the counting, no substatement need be executed. To indicate the omission of the substatement, a single semicolon has been included after the for-loop definition.

For-loops are very important in C programming. As I have stated, they are probably the most often used iteration-type instruction. C programmers do, however, have two other looping instructions at their disposal—the while-loop and the do while-loop (commonly called the do-loop). Although these instructions have very similar names, there are some very important differences between them. We will discuss each instruction individually and contrast them with the for-loop.

```
while( expression )
        substatement;
```

Unlike the for-loop, the while-loop only has one expression in the initial instruction definition. This expression is equivalent to the second expression in a for-loop. The substatement is executed as long as the expression remains true. As with the for-loop's second expression, if the expression is not true (it evaluates to zero), the substatement is not performed and execution begins at the next instruction after the substatement.

A while-loop's substatement has the same function as the for-loop's substatement. If you want a series of substatements to be controlled by the while-loop, simply place the open brace before the first substatement and the closing brace after the last substatement.

If you want a while-loop to operate similarly to the for-loop, it should look something like the next example. Please note: unlike the expressions in the for-loop, the expression in the while-loop is not optional. Leaving it out causes a compiler error. If you want to make a while-loop into a forever loop, simply insert any non-zero numeral as the expression.

```
        expression 1;        /* make a "for-loop" out of a while-
loop*/
        while( expression 2) {
                substatement;
                expression 3;
        }
```

Why use a while-loop when a for-loop can do the same thing? Good question. The best answer I can give is "style." When we discussed the for-loop, I mentioned the "normal" use of expressions that made up the definition. The first was for initialization, and the third for incrementing. When you see a for-loop, it is usually safe to assume that something is being performed in an ascending or descending sequence. Now suppose a looping task must be performed, but values do not necessarily change after each iteration. Using a while-loop in this case would be better programming style because it does not elicit the "sequential loop" impression that a for-loop usually indicates.

Finally, we come to the last loop instruction, the do while-loop. Unlike the others, do-loops are for occasions when the substatement must be executed at least once before the expression is evaluated.

```
        do
                substatement;
        while( expression ) ;
```

As in the other examples, the expression determines whether the loop continues or ends. However, unlike the other loop instructions we have discussed, the expression is not evaluated until *after* the substatement is executed. This means the substatement is executed at least once. (When several substatements are required, the open brace is placed after the "do," and the closing brace is placed before the "while.") As in the while-loop, the expression cannot be omitted without causing compilation errors. Additionally, notice that a semicolon follows the "while" part of the instruction.

Again, the question arises: "Why use the do-loop when the for- or while-loop can be used?" Again the only answer: "style." Despite the major difference between the do-loop and the other loops, it is used infrequently. Most programming tasks just do

not fit the do-loop scenario. For an example of when you might use a do-loop, look at this function.

```
/* This function will continuously get keys from the keyboard */
/* until the key requested has been entered                   */

wait_for( my_key )
short my_key;          /* Key to wait for */
{
short key;             /* Temporary holding key */

        do {           /* Begin looping */
        key = getchar();
        /* get a key from keyboard */
        } while( key != my_key);
        /* check for a match */
}
```

This function waits for a user to type the requested key. Since we know we must get at least one key, we can use the do-loop. Although the braces are unnecessary in this example, I like to use them with do-loops to "tie" instructions together (since two words are required). After a key has been returned from the keyboard function "getchar," the do-loop evaluates to see if the loop should continue. The loop ends, and our function returns only when the key is equal to the requested value.

That completes the discussion of looping instructions available in C. With these definitions in hand, you are now ready for any looping occasion...or almost ready. I should mention a couple of statements and hints regarding loops because sometimes you don't want to wait for the loop evaluation to get you out of a loop.

The "return" statement is probably the most obvious way to get out of a loop. (It also gets you out of the function in which the loop is located.) During the course of a loop, if you determine that it is time for the entire function to end, execute a return. Here is an example.

```
/* This functions sets the values in an array to zero until */
/* a zero or the max index of the array (99) is encountered */

set_zero( my_array )
short my_array();      /* array of values */
{
short var;             /* a counting variable */
for(var = 0; var < 100; var++) {   /* loop through the array */
        if ( my_array(var) == 0 )          /* is this a zero */
return;                                    /* yes, then return */
my_array(var) = 0;                 /* no, then set to zero */
        }
}
```

Sometimes you don't want to return from the function; you just want to terminate the loop. The "break" statement was designed for this. When a break statement is encountered inside a looping instruction, execution "jumps" to the next statement after the loop. Using our last example, suppose we want to return the index where zero is encountered, or return 100 if it is not found. (We know 100 means it was not found because we are checking for values less than 100. When the variable is equal to 100, the loop terminates automatically.) The function looks like this.

```
/* This functions sets the values in an array to zero until */
/* a zero or the max index of the array (99) is encountered */
/* The final index value is returned                        */
```

```
short set_zero( my_array )      /* function returns a short*/
short my_array();               /* array of values */
{
short var;                      /* a counting variable*/

for(var = 0; var < 100; var++) {  /* loop through the array */
if ( my_array(var) == 0 )       /* is this a zero */
break;                          /* yes, then terminate */
my_array(var) = 0;              /* no, then set to zero */
        }
        return(var);
/* return the index value*/
}
```

Occasionally, while executing inside a loop, you may want to skip the rest of the statements without terminating the loop. Although you can encase the remaining statements in a block controlled by an "if," the "continue" statement may be more straightforward. When the continue is encountered, control "jumps" to the bottom of the loop, but does not "break" out of the loop.

Referring back to our for-loop definition, we know that expression 3 is the next instruction executed. For while- and do-loops, the expression that determines whether the loop should continue is executed next. In our running example, suppose that while inside the loop, only values less than 10 should be set to zero. Now our function looks like this example.

```
/* This functions sets the values in an array that are */
/* less than 10 to zero until          */
/* a zero or the max index of the array (99) is encountered */
/* The final index value is returned          */

short set_zero( my_array )  /* function returns a short*/
short my_array();           /* array of values */
{
short var;                  /* a counting variable*/
for(var = 0; var < 100; var++) {  /* loop through the array */
if ( my_array(var) > 10 )   /* if outside the range */
        continue;           /* continue the loop */
if ( my_array(var) == 0 )   /* is this a zero */
        break;              /* yes, then terminate */
my_array(var) = 0;          /* no, then set to zero */
        }
        return(var);
/* return the index value*/
}
```

(continued)

Finally, we come to the infamous "goto." Goto instructions are jump instructions and can send control anywhere inside a function. The syntax of the goto follows this paragraph. Although this syntax definition shows that the label comes after the goto statement, the label can occur anywhere inside the function the goto is located in. Labels must be unique to a function. You cannot have two labels named "abc" in the same function, but a label "abc" can occur in more than one function.

```
        ....                /* indicating previous code */
        goto label;         /* jump to the desired label */
        ....                /* more code          */
label:              /* name of label followed by colon*/
        ....                /* and more code      */
```

I will not sit and argue about the implications of using a goto instruction. Yes, the goto usually leads to the dreaded unstructured code, and, in most instances, it can be avoided. Goto's, though, can be a highly efficient way of controlling program flow through deeply nested loops or error conditions.

Consider a routine that performs several routine file operations, including reading and writing. When an error occurs somewhere along the route, the user is usually notified that an error has occurred, the file is closed (and perhaps deleted), and an error code is returned to the calling function. Although you can avoid a goto by including these instructions at each hazard point, it is sometimes more convenient and efficient to place the error instructions in a location within the function. This way all the other hazard locations can simply "jump" to those instructions using a goto. My advice about using goto instructions is simple: If you think you need one, use it. Don't fret over it.

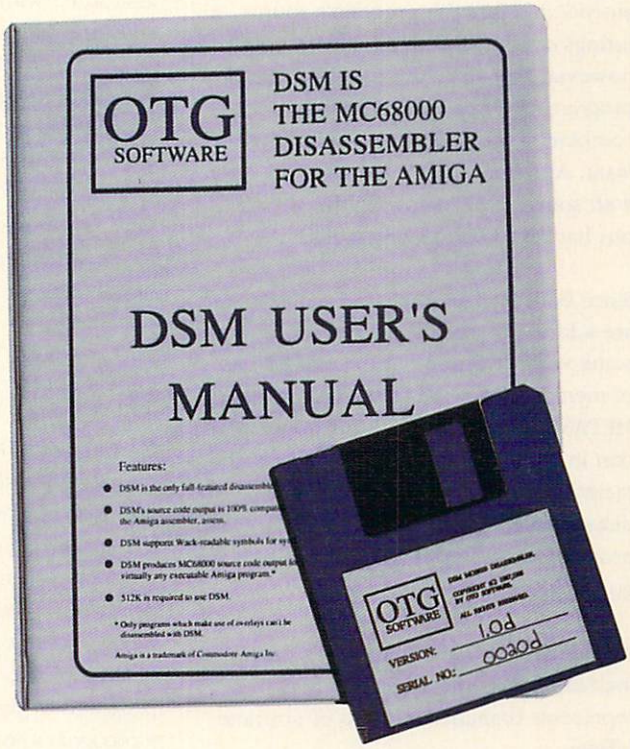
That about does it for this month. If you still have questions about loops or the other supporting statements, consult a good textbook or visit your local bulletin board. Try writing a program that uses the instructions we have discussed. Remember, you can learn a lot by experimenting.

•AC•

OTG SOFTWARE'S DSM, A 68000 DISASSEMBLER

by Gerald Hull

A "disassembler" is the functional inverse of an assembler. An assembler takes assembly language source and produces machine code; a disassembler takes an executable program and produces an assembler listing. Obviously a very powerful tool for anyone involved in programming. With the release of OTG Software's DSM MC68000 disassembler, that power is now fully available for the Amiga.



A Look At The Competition

DSM is not the first disassembly product available for the Amiga—there are at least four others (perhaps more). Two public domain products, DISASSEM and DIS, have been released on Fred Fish disks 27 and 128, respectively. In addition, disassembling capabilities are present in both Metadigm's METASCOPE debugger and Abacus's ASSEMPRO assembler.

Nonetheless, at least as far as these four are concerned, OTG seems correct in claiming that "DSM is the only full-featured disassembler for the Amiga." Like DSM, the public domain products are "program-based": you feed them a program, and they spit out a listing.

In contrast, METASCOPE and ASSEMPRO are both "address-based," meaning they first load the program, and then let you place a window on the area in memory where the program is located. To produce a listing, you then tell the disassembler the starting and ending addresses of the code you want disassembled. One advantage of the address-based approach is that it lets you look at the Amiga ROM code and figure out how to take best advantage of the operating system.

The evaluation of these products depends on the purpose you have in mind. They all produce 68000 mnemonics and directives. If you are only interested in the details of what's going in a program, then offsets, line numbers, and hexadecimal representations of the machine code are quite useful:

```
0000   93c8        1    sub.l   a0,a1
0002   2009        2    move.l  a1,d0
0004   6704        3    beq.s   000a
0006   d2fc  0010  4    adda.w  #$10,a1
000a   4e75        5    rts
```

However, if your desire is just to modify and easily reassemble the code, all that extra information gets in the way. You have to go through the listing, deleting and correcting, to get something an assembler accepts:

(continued)

```
sub.l    a0,a1
move.l   a1,d0
beq.s    L1
adda.w   #$10,a1
L1       rts
```

DSM particularly shines at correcting; none of the others measure up to it. The DISASSEM public domain program does not allow you to suppress offsets or hex code, and is pretty buggy as well. The DIS program lets you suppress everything but the mnemonics, and to some extent, is able to generate appropriate labels.

In other cases, however, DIS generates branches to absolute locations—BRA 000110—that need to be carefully amended before the program's source listings can be assembled. Please note, however, that since these public domain programs are free, we can hardly complain if they fail to do everything we want. And both DISASSEM and DIS come with source code, allowing the industrious hacker to correct any faults.

Since both METASCOPE and ASSEMPRO are address-based, the user has to do some work to figure out the exact areas of memory that need to be disassembled. METASCOPE provides all this information in a "Hunk Window." However, the memory locations you specify must be disassembled as either code or data. (The program has no machinery for distinguishing between the two).

And finally, METASCOPE's disassembly includes the memory location and represents branches in terms of absolute offsets:

```
c3564c    beq.l   *+$72
```

All such references need to be carefully replaced with labels before making any attempt to alter the code.

Once you get past a sometimes infuriating user interface, the ASSEMPRO "Reassembler" produces a relatively decent assembler-ready listing. All extraneous information is suppressed, and appropriate labels are generated. Unfortunately, the program gives the user no assistance in determining what addresses contain the program as a whole (which can be in a number of separate hunks). Furthermore, sometimes its JSRs are to illegal program counter offsets: for example, JSR $C6FAAC(PC).

A Power So Great

In contrast to these four disassemblers, OTG's DSM makes it very simple to generate well-labeled (and hence, easily-modified), assembler-ready code. At this point, the question is, "What's that good for?" The smart-alecky response is, "If you have to ask, you don't need the program." For more adequate reply, we must look at some of the difficult issues associated with copy protection and software piracy.

People of my generation took (and still take) a special delight in the verbal wizardry and profound wit of a comedic quartet called "The Firesign Theater." One line in particular frequently occurs to me when I'm confronted with the burgeoning capabilities of technology: "A power so great it can only be used for good or evil." In a nutshell, that's what DSM provides.

Starting with the more innocent cases, we often find that software designed for the mass market does not fully address the special circumstances of its users. For instance, you may have three floppy drives, but the file requestor only recognizes two. If so, you may find it very helpful to use DSM to customize programs to your own needs by inserting a recognition of DF2:.

A second example is a bit weightier. A friend of mine bought a SCSI daughter board for his memory expansion unit on the manufacturer's promise that drivers would be provided for OMTI controllers. The company (which shall go nameless) has since welshed on that promise. With DSM, my friend can disassemble the driver they did provide and rewrite it for the OMTI protocol.

However, one of the more useful applications of DSM takes us into deep moral waters. The DSM manual explicitly recognizes this function: "copy protection features could be eliminated thus allowing the user to make backup copies of a program, or to place a program onto a hard disk drive."

Now the manual carefully notes that this use of DSM may violate some software licensing agreements, advises users to "assure yourself that your actions are legal," and disclaims responsibility for any misuse of the program. Without pretending to have any legal expertise, I see nothing ethically wrong in defeating copy protection on software you legitimately own.

What is definitely wrong is any subsequent transfer of de-protected software that deprives its developers rightful compensation. Unfortunately, in this often unfair world, it is not always possible to prevent people from making immoral choices. However, in my eyes, the legitimate benefits of a program like DSM outweigh its power for doing ill. In any event, concerned developers can protect themselves from misuse of DSM.

DSM In Action

Although DSM is much better than the other four programs for generating useful assembler source, it nonetheless runs up against certain inherent limitations. The program I used it on is 69000 bytes long. The assembler source code generated by DSM, however, is 369000 bytes long, an over five-fold increase. The manual indicates that the source files may be "5 to 15 times the size of the original binary file."

Since many programs are significantly larger than the one I worked with, absolutely gigantic source files are an unavoidable liability. Printed out, that 369000 byte file resulted in hardcopy an inch thick. If you don't have much memory, your only recourse (as the manual suggests) is to direct DSM's output to disk.

You also confront the problem of editing such a large source file. If you are strapped for memory, your only recourse may be to use DSM's size option which

allows you to specify a maximum file size. You do so by indicating the number of block, starting with a 25 block minimum. (The manual erroneously states that a block is 256 bytes long; the correct value is 512.)

This option prompts DSM to break the source down into multiple text files, none of which are larger than the size you specify. When I used a maximum of 50 blocks, however, the result was 142 files, ranging in size from 805 to 25162 bytes. These files are organized according to segments, and all files constituting a single segment are tied together with INCLUDEs.

Unfortunately, DSM analyzed the program into 136 separate segments mandating a rather complicated reassembly process. According to David Hankins, author of DSM and president of OTG Software, this segmentation is determined by the compiler (Lattice in this case). DSM leaves the segmentation alone; tampering with it might wreck the program.

Since I have a lot of memory, I worked with the single 369k file. Although this saved some inconvenience, ED is very slow with files that large. As you might imagine, DSM also takes some time to generate these source files. Using the "expert system" option (more on this in a moment), generation took about 10 minutes.

This is hardly unreasonable, however, since it takes about that much time for an assembler to "reconstitute" the program. By design, DSM targets its output for the Metacomco ASSEM assembler, befitting ASSEM's status as the "default standard" for Amiga assembling. With MCC version 11.0, reassembly took thirteen minutes.

For comparison, I also tried a beta release of version 2.0 of Wesley Howe's CAPE assembler (marketed by Inovatronics). Howe has apparently been doing his homework; his program finished off the huge source file in a mere two and a half minutes.

To get a good taste of its capabilities, I used DSM to remove the "manual look-up" protection on a program I own. My initial "code breaking" venture was a success. I now can use the program without compulsory browsing in the manual's deathless prose. Interestingly enough, the reconstituted version of the program is some 8k smaller than the original. This resulted from hunk consolidation by DSM, and the fact that I relinked using the SMALLCODE and SMALLDATA options of BLINK.

Load File Bloat

Other experiments with DSM indicated that the reconsituted program is much more likely to be larger than the original. For example, if you apply DSM to itself and reassemble, the resulting program is 8k larger than the original. At first, I thought this reflected a bug or weakness in the program.

I discussed this problem with OTG's Hankins (who is accessible to users on COMPUSERVE, BIX and PLINK), and he concurred that his goal in writing the program "was to develop a disassembler which would give you source code that... [would] produce an exact copy of the original executable program." However, the different capabilities and limitations of Amiga compilers, assemblers, linkers, and loaders conspire to make that goal unreachable.

The size gain in the reconstituted version of DSM results from a trick used by the Manx compiler (Lattice does the same thing). A loadable Amiga program consists of a number of hunks of CODE, initialized DATA, or uninitialized data (BSS). However, the size of each hunk is specified twice. The first specification is in a header block which the loader uses to determine how much memory must be allocated for the program as a whole.

The size is given a second time by a number at the start of each constituent hunk, indicating how much loadable stuff of the particular type follows. If the space already allocated for a CODE or DATA hunk by the header block is larger than this second value, the program has effectively allocated for itself an uninitialized data area without explicit recourse to a BBS hunk. This reduces the size of the load file, making compiler owners happy.

Because the Metacomco ASSEM assembler doesn't know this trick the load file it creates bloats up to contain that previously concealed space. Why doesn't DSM put it where it belongs in a BSS hunk? The answer is that AmigaDOS "scatterloads" hunks wherever it finds unused memory. The rest of the reconstituted program, unaware that space had moved, would start chucking data into another task's domain which calls down the wrath of the Guru.

Challenging The Expert

Most disassemblers run sequentially through code, translating each word or group of words as it comes. By default, everything is assumed to be code. If something makes no sense as code, the disassembler either rejects it as an "illegal

(continued)

instruction," or, if the program has a modicum of intelligence, deciphers it as data. When this occurs, it often takes a number of words before the disassembler gets back on track.

Of course, if an executable instruction is represented as a series of byte constants, it still reassembles and runs. For example, if

```
sub.l    a0,a1
```

is rendered as

```
dc.b     $93,$c8
```

the reconstituted program works just fine. Indeed, in some cases DSM deliberately uses the byte constant format for instructions that might otherwise be incorrectly reassembled. (In such cases, the desired instruction is given in the comment field.)

In general, the representation of instructions as data precludes the user from understanding what the program is doing (and hence from adapting it to other purposes). In contrast to the "code-default" approach, DSM attempts to follow the possible paths of control flow in the program it's disassembling. Only segments reachable this way are construed as code, and if any words fail to fit that interpretation, the entire segment is recast as data.

The result of this "data-default" approach is much more reliable disassembly. Of the four competitors, only the Abacus product uses similar strategy. Here DSM offers a great advantage by allowing the user to invoke an "expert system" option. This expertise involves intelligent guesses of where branches may logically occur in a disassembled program. The results are a much more thorough ferreting out and accurate translation of code segments.

Even experts can be confounded on occasion, though, and DSM's artificial intelligence is no exception. Some control flows simply cannot be followed. The manual states that:

"this happens when code statement references are made exclusively with the following addressing modes: address register indirect, address register indirect with displacement, address register indirect with index, or program counter with index."

With the conventional programming techniques implicit in most high-level languages, such addressing modes occur rather infrequently (aside from the occasional jump table). However, "threaded code" languages like FORTH systematically use jump tables for program control. My guess is that DSM's expert system would encounter greater difficulty disassembling FORTH code.

Finally, a perverse programmer, perhaps in a deliberate attempt to foil products like DSM, can introduce code sequences designed to baffle disassembly:

```
sub.l    a1,a1
move.l   a1,d0
beq.s    L1
dc.w     $face
L1       rts
```

Although superficially similar to the code fragment given earlier, this segment represents a "poison pill" for DSM's expert system. Indeed, it induces an earlier version of the program to abort. (This has been corrected in the official release.)

Since SUB.L A1,A1 effectively clears the A1 register, the branch that follows is always taken. However, DSM's "expert" isn't smart enough to realize this, and because it finds a non-instruction following the branch ($FACE), it assumes the entire segment must be data. To accommodate such cases, DSM provides an option that generates the relative offset of every line as a comment in the disassembled program. The drawback is that this operation doubles the size of the listing produced.

A correlated option allows you to specify an "offset file" in addition to the load file of the program you are disassembling. In this file, you put the addresses of any data statements you suspect actually constitute code, and thereby augment DSM's expertise. By successive iterations of this process, the manual says, "DSM should enable you to arrive at...a completely accurate translation of the target program into MC68000 code and data statements." As frosting on the cake, the program automatically incorporates debug information into the disassembled listing.

Some limitations of DSM need be mentioned. The manual notes that the program is currently unable to handle 68010, 68020, and 68881 instructions and does not disassemble programs with overlays. These capabilities are listed as "future enhancements." Finally, the current version of DSM cannot disassemble object (.O or .OBJ) files; that is, it only works on executable programs.

Conclusion

It didn't take much experimentation with the OTG disassembler for me to wonder how I had ever gotten along without it. DSM is an extremely powerful tool that every serious programmer should own. Although there are some constraints and limitations, most of them are unavoidable in this type of program. No similar product I am aware of can compare with DSM in power, effectiveness, and ease of use.

The 1.0d release of DSM retails for $67.50 and consists of single, unprotected disk and a 60-page manual in a three-ring binder. The projected upgrade cost is $10.00 to $20.00, depending upon degree of enhancement. The manual is well-written, logically organized, and chock full of useful information. Along with the DSM disassembler, the disk contains a utility called ATEM (modelled after Commodore's ATOM) that allows the user to control the type of memory program segments get loaded into.

•AC•

DSM
$67.50
OTG Software
200 West 7th Street, Suite 618
Fort Worth, TX 76102
Available at selected Amiga dealerships

by John Steiner

Bug Bytes

The Bugs & Upgrades Column

Here's a bit of interesting information for registered owner of Superbase Personal and Superbase Professional. On August 15, 1988, Precision, Inc. of Irving, Texas took over the U.S. support of all versions of the two programs. Progressive Peripherals & Software in Denver previously handled support.

Precision has also announced that the dongle copy protection has been removed from both current packages.

A new product, Superbase II, has also been introduced. The database is targeted a market between the markets covered by the two previous Superbase products. Upgrades from one version to another cost only the price difference between the two packages, plus $10.00. Registered owners of either Superbase product can now obtain customer support from:

Precision, Inc.
8404 Sterling St. Suite A
Irving, TX 75063
(214) 929-4888

Micro-Systems Software's advanced word processing package, excellence!, is quickly becoming a popular word processor. One common complaint about the program, however, is that it is rather slow. Harv Laser of People Link posted a file from Usenet with some suggested speed improvements and a couple of bug reports.

- To improve speed while using excellence!: Reduce the number of screen colors to two.
- Don't use the interactive spelling checker.
- Load the dictionary into RAM.
- Keep the glossary small.

When you enter text, enter at the bottom of the document.

If you have a single drive system, you must swap disks when accessing various program overlays. If you have one or more megabytes of RAM , you can order a version of the software without overlays. If you wish to order the one megabyte version. They require an return authorization, so be sure to contact them first. You must return your original disk, and the upgrade costs $14.95.

Now for the bugs:

Pressing Tab followed by BackTab causes a crash. This bug has been corrected in the latest version.

The continuous spell checker does not check misspellings of two characters or less, but the normal spell checker does. This bug was verified by Micro-Systems Software.

If you have only one disk drive, the program does not find the excellence!.prefs file. Therefore, your preferences are not loaded the next time the program is accessed. This problem prevents you from loading your user dictionary and glossary file if you have only one drive.

For users with a single disk drive, here's a solution to the above problem. The problem is that the Workbench disk is in the drive when the program checks for the excellence!.prefs file. When the program does not find the file on the Workbench disk, it does not ask for the excellence! disk. The program assumes that you don't have a .prefs file (nor a .gloss and .udict file). You need to make these files available on a device named excellence!:.

One way to fix this is to copy your disk, (never change the original) and to name the new disk excellence: (Do not use the exclamation point in the name). Next, copy the stack program from the C: directory on your Workbench disk to the newly created excellence: disk, and copy the following 8 lines to a file called startup on the excellence: disk. To run excellence!, just execute the startup file.

```
copy excellence:excellence!.udict ram:
copy excellence:excellence!.gloss ram:
copy excellence:cmd.gloss ram:
copy excellence:excellence!.prefs ram:
assign excellence!: ram:
assign doctools: excellence:
excellence:stack 16000
run excellence:excellence!
```

Only one .gloss file is needed. Specify its name in the preference requester for excellence! To make changes to your .prefs, .gloss, or .udict files, you must copy those files from ram: back to excellence: after you finish with excellence! You can also create an execute file for this.

Micro-Systems Technical Support
4301-18 Oak Circle
Boca Raton, FL 33431
(407) 790-0772

When editing a text file from within a word processor, you surely have noticed a carriage return at the end of each line. To be compatible with most word processors, your document must have only a hard return at the end of each paragraph.

Here is a way to replace the hard returns at the end of each line with a space character. This method works only with a word processing package, such as WordPerfect, that lets you search and replace the hard return character.

Begin by going through the text file to make sure there is an extra carriage return between all paragraphs. If there is only a single hard return at the end of each paragraph, press Return to produce double spacing between paragraphs.

Do a search and replace to change all double hard returns to "&&" or some pair of characters that never occurs together in your document.

Next, search and replace all hard returns with " ", the space character.

Finally, search and replace the "&&" with a hard return. This leaves hard returns only where there is a blank line.

Thank you Mike Scalora of WordPerfect for this tip. If your word processor doesn't support search and replace of hard returns, I have written a program that reads a text file and replaces each hard return with a space. The program, HRTstrip, has been uploaded to the People Link software library. If you would like this program and cannot get it through People Link, contact me at Amazing Computing, and I will let you know how to obtain a copy.

Mike Scalora of WordPerfect also suggests that you can increase speed when moving through a large document by enlarging the default edit buffer. The default buffer size is 16K, but you can make this larger if you have more than 512K.

If you begin WordPerfect from the CLI, you can change the edit buffer size by typing WP -w 100 to increase the edit buffer to 100 Kbytes. If you start WordPerfect from the Workbench, you must modify the Tool Types insertion bar in the WordPerfect Workbench icon. To modify the edit buffer size, click on the WordPerfect icon to highlight it (again, on a copy of your WordPerfect disk), and choose Info from the Workbench menu bar. When the Info window opens, look at the Tool Types insertion bar, click on the ADD button, and then click on the insertion bar. Now just enter WORK AREA = 100 and click on the Save button.

As you can see from the above examples, 100 represents the buffer size in kilobytes. If you want to use a different buffer size, just type the number in kilobytes. Increasing the edit buffer size dramatically decreases the time it takes to move the cursor from the top to the bottom of a large document (especially if your file is stored on a floppy disk).

A WordPerfect update, dated August 10, 1988, repairs many spell and spell utility bugs and a bug in the display code. Registered owners can get upgrade information by calling WordPerfect, Inc. at the order line listed below or by calling WP technical support.

WordPerfect, Inc.
1555 N. Technology Way
Orem UT 84057
1-800-321-4566

Microsmiths Software's premier text editor, TxEd Plus, has been upgraded to version 2.01. According to a company spokesperson, all registered 2.0 owners will be sent 2.02 "in a few weeks." If you have registered your software, you may have already received this upgrade.

Version 2.02 has the SAVE AS keyboard shortcut, as well as other bug fixes.

Microsmiths, Inc.
Box 561
Cambridge, MA 02140
(617) 354-1224

Impulse! Software will release version 3.0 of their 3-D rendering program, Turbo Silver. The major upgrade from Turbo Silver 2.0 to 3.0 is the addition of "super requesters." Previously, all the attributes were set on several different screens of sliders—colors on one screen, position on another, size on another, attributes on another, and so on. Turbo Silver 3.0 distills all those sliders down to two or three large requesters that fill about 3/4ths of the screen.

So when you are doing camera settings, one requester, complete with sliders and string input requesters, has all possible settings. You can see the relationship between the different settings and set or modify them all at once. The same technique is used for the "global" settings (ambient light, etc), and objects. Certainly a major improvement.

A "ground" object that can be plain, colored, or IFF-mapped has also been added, along with several other time-saving improvements. The manual is also being entirely re-written. 3.0 should be ready to ship as you are reading this.

Impulse!
6879 Shingle Creek
Pkwy Suite 112
Minneapolis, MN 55430

That's all for this month. If you have any workarounds or bugs to report, or if you know of any upgrades to commercial software, notify me by writing to:

John Steiner
c/o Amazing Computing
P.O. Box 869
Fall River, MA 02722

...or leave EMail to
Publisher on People Link or
73075,1735 on CompuServe

•AC•

HAM & AmigaBASIC

by Bryan Catley

Even though it is easy to use any 32 of the 4096 colors available on the Amiga from AmigaBASIC programs, HAM (Hold And Modify) mode has always been held as a somewhat mystical mode of operation. Primarily because this mode allows all 4096 colors on the screen at one time, but also because this mode is simply not available to BASIC programmers! And while many people not fully understand it, most do realize that any one pixel's color is usually determined by its neighbor.

In this tutorial we'll try to take some of the mystery away and explain how HAM works, and how you can create a screen with many of the 4096 colors with an AmigaBASIC program! However, before we get started, I must make an acknowledgment.

The idea for this article came from an example program included with Absoft's AC/BASIC compiler, Release 1.3. The program included here is based directly on that example. The comments have been modified somewhat, and some text is written on the HAM screen. The display has also been changed, but we'll discuss this in depth a little later. For now, let's just get a better understanding of how HAM colors are displayed on the screen.

Amiga Color Registers

The first things we'll look at are the color registers the Amiga uses to generate the various colors displayed on the screen. There are 32 color registers, and the number available at any one time is determined by the depth of the screen. The following table provides the exact relationships:

Basic Screen Depth	Number of Color Registers Available
1	2
2	4
3	8
4	16
5	32

In case you haven't realized it, a color register is known as a PALETTE in AmigaBASIC.

Why 4096 colors? Why not more, or less? Well, each color register is composed of 16 bits made up as follows:

0000 rrrr gggg bbbb

The first four bits are unused, the remaining 12 are used to provide four bits of red, four bits of green, and four bits of blue color information respectively. The combination of these three pieces of r, g, and b information provide the final color, which is displayed on the screen in each pixel for which that color register has been specified.

Now, if you are at all familiar with the binary numbering system, you know that four bits can represent from 0 to 15, for 16 different values (or variations). This means that four bits each of red, green, and blue information can represent 16 x 16 x 16 different colors for a grand total of 4096 colors per color register (or palette)!

Consider the possibility of expanding each of the r, g, and b variations to five bits. This would provide for 32 variations, or 32 x 32 x 32 different values for a total of 32,768 colors! Maybe sometime in the future?

Bit Planes

Now that we understand how color registers work, it's time to see why we are (normally) restricted to 2, 4, 8, 16, or 32 of those 4096 colors at any one time. To do this, we must go back to the screen depth parameter which must be specified each time a custom screen is defined in an AmigaBASIC program. (If it isn't, the default Workbench screen is used, which has a depth of two).

When you define a screen and specify a depth, you are actually specifying the number of bit planes to be allocated to this screen. A bit plane is an area of memory in which each bit represents a single pixel on the screen.

If you only have one bit plane, then each bit can only have a value of 0 or 1, which means you only have access to color registers (or palettes) 0 and 1. However, should you specify a depth of two, you will have two bit planes for the screen and each pixel on the screen will be represented by two bits (one in each bit plane). This means each pixel may now be represented by values of 00, 01, 10, and 11; or 0 to 3. These values indicate the color register used to display the corresponding screen pixel.

If you follow this logic through bit planes three to five, you will be able to see how the above table was arrived at. Each new bit plane simply doubles the number of available color registers!

Remember that each new bit plane uses a substantial amount of memory; this is why many programs do not automatically use all 32 color registers.

Okay, now we know how bit planes are used to determine which color registers are used to display each pixel, it's time to see how all this relates to Hold And Modify.

Hold And Modify

The first thing to realize about HAM mode is that it requires a screen with six bit planes, something that AmigaBASIC does not allow. This is why HAM screens are not normally available from within AmigaBASIC programs. Secondly, six bit planes provide a theoretical maximum of 64 colors, a far cry from the maximum of 4096 available when using HAM. Why the difference?

Well, in HAM two of those six bit planes take on new meanings, and these new meanings allow up to 4096 colors on the screen at any one time! When using HAM, bit planes one to four determine which color register is to be used (0 - 15), while bit planes five and six tell the system how to "Hold" and "Modify" those colors!

This means if a pixel is drawn specifying color registers 0 - 15, the pixel is drawn in that color, as normal. However, if color register numbers 16 - 63 are used bit planes five and six come into play. Whenever bit planes five or six are non-zero (any color register specification greater than 15), the color from the preceding pixel is reproduced and modified based on the contents of bit planes five and six.

The actual modification is based on the following table:

Bit Planes 6-5	Color Modification
00	Color displayed as normal
01	Hold Red and Green; Modify Blue
10	Hold Blue and Green; Modify Red
11	Hold Blue and Red; Modify Green

The modification itself is based on the contents of the color register specified in bit planes 4 to 1. That is, two of the three color variables are "held" from the preceding pixel, while the third variable is "modified" based on the current color register specification (in bit planes 4 - 1). As can be seen, this allows almost perfect "shading" to be achieved while coloring objects.

An AmigaBASIC HAM Example

Since AmigaBASIC will not support a six bit plane screen you must circumvent the normal screen/window commands to display HAM graphics. This is not too difficult because the NewScreen and NewWindow structures used by C, assembler, and other languages may be easily reproduced in AmigaBASIC using integer arrays.

Once these structures are set up, call the appropriate ROM Kernel routines to set up the custom screen and window for use. Remember, when this is done, AmigaBASIC has no knowledge whatsoever of the screen or window. This means that normal window operations (including mouse and keyboard input) may no longer be used, and that the customary AmigaBASIC menus are also no longer available. It's ALL up to you!

The accompanying example program explains how to do this. As previously mentioned, it is based on the HAM example Absoft provides with their AC/BASIC compiler Release 1.3. Some comments have been modified, some text is displayed on the screen, and the actual HAM display is created using a variation on the original program's technique.

The original program used a random number to determine each pixel's color with a statement: pen% = INT(RND * 31 + 1). This sets the pen to a random value of 0 to 31. Now, quickly, will this work? Unfortunately the resulting display is not the greatest HAM example! If you've been following, the reason why is fairly obvious.

Restricting yourself to palettes 0 - 31 means you will only see the HAM effect when the random number generator provides a number between 16 and 31. At all other times the standard color, as specified, will be used! So, if you own AC/BASIC Release 1.3, the first thing you want to do is change both these statements to pen% = INT(RND * 63 + 1). Now all six bit planes come into play, and the difference in the resulting display is quite startling!

In the accompanying example, things have been carried a step further by starting with pen% = 0, and then by continually adding one each time through the loop. (The value of pen% automatically returns to zero when 63 is reached.) This variation was arrived at after discussions about the program with the folks at Absoft, and it provides a rich selection of HAM displays, depending on when and where pen% is incremented!

Type in the program, save it, and run it. For variations, try changing the way pen% is incremented. Reset it to zero at the

beginning of each loop; allow it to keep cycling through each loop; reset it to zero at the start of the first loop only; reset it to zero at the start of the second loop only; or any other approach you can dream up (like incrementing by two rather than one). The variations in patterns you end up with will be substantial and impressive.

Please note that you don't need AC/BASIC to run this program, but it certainly speeds things up! Under the AmigaBASIC interpreter, this program takes about six and one half minutes to complete its display on the screen. However, if you compile and then run it, it takes just one minute and 50 seconds to complete the display! (The listing recommends options to use if you compile the program.)

Remember, AmigaBASIC knows nothing of the screen and window you create with this program, so neither mouse clicks nor keyboard presses are recognized. Further, no menus are present and you cannot use the normal CTRL-C to interrupt the program once it starts to execute, you must let it finish or reboot the machine. This is why, once the display has completed, the program waits for a given period of time and then terminates automatically. It is also why error conditions are announced through the Amiga's speech facilities; there are no known windows to write to!

These restrictions tend to make using a HAM screen from within AmigaBASIC rather impractical, (except for demonstration purposes). However, I'm sure it is just a matter of time before someone invents a method of controlling these restrictions.

For starters, it would help if Commodore and Microsoft released a new version of AmigaBASIC which would allow you to specify six bit planes when defining screens; once AmigaBASIC can recognize a HAM screen (or HalfBright - which also requires six bit-planes), all sorts of things will become relatively easy to accomplish!

```
'*********************************************save*********
'*
'* This program is a modification of an example program which
'* is distributed with Absoft Corporation's AC/BASIC
'* Compiler Release 1.3.
'*
'* First, the program does some minor work to get the machine
'* in Hold AND Modify mode.  After that, it paints the screen
'* full of HAM colors and then undoes all that it did.  If an
'* error occurs, it cleans up any mess it may have made,
'* announces the error (using SAY as we don't know the state
'* the screen/window system if trouble arises somewhere), and
'* bails out.
'*
'* Note: To fully understand Hold And Modify graphics,
'*       you should examine the ROM Kernel Manual:
'*       Libraries and Devices publication.  This manual
'*       discusses the graphics library in detail.  An
'*       understanding of Intution is also essential.
'*       See the Intuition manual.
'*
'* Another Note:  This program is reasonably
'*       unsophisticated insofar AS no routines check for
'*       keyboard/mouse activity in the active window/screen.
'*       This means that trying to dump the program with
'*       control-C won't work.
'*************************************************************
```



```
'*
'* Use compile time option T with the AC/BASIC compiler.
'*
'*************************************************************
'*
'* Modification History:
'*
'* 29 Mar 1988  Original version prepared for AC/BASIC
'*              v1.3 release                                CCG
'* The following modifications were made for publication
'* in Amazing Computing:
'* 11 Jul 1988  Introductory text added                     BDC
'* 11 Jul 1988  Pen number for displaying HAM colors changed
'*              to a cycling number from 0 - 63             BDC
'*************************************************************

'_____________________________
'- Various setup stuff                                    -
'_____________________________

DIM NewScreen%(15)
DIM NewWindow%(23)

'- define all the external stuff required -

DECLARE FUNCTION OpenScreen&() LIBRARY
DECLARE FUNCTION OpenWindow&() LIBRARY
DECLARE FUNCTION ViewPortAddress&() LIBRARY

LIBRARY "graphics.library"
LIBRARY "dos.library"
LIBRARY "intuition.library"

'_____________________________
'- Initialization code.  Get the machine in HAM mode -
'_____________________________

'- set up the screen definition -
```

```
' We create our own data structures to create
' NewWindow/NewScreen definitions.  These are then
' passed into ROM Kernel routines to create windows
' and screens that Amiga BASIC can't create.
'
' Note: Since Amiga Basic doesn't support structures,
' we use arrays to achieve the same results.

NewScreen%(2) = 320    '- Screen width
NewScreen%(3) = 200    '- Screen height
NewScreen%(4) = 6      '- Screen depth
NewScreen%(6) = &H800  '- Hold And Modify mode
NewScreen%(7) = &HF    '- Custom screen

'- set up the window definition -

NewWindow%(2)  = 320  '- Window width
NewWindow%(3)  = 200  '- Window height
NewWindow%(4)  = &H1  '- DetailPen, BlockPen
NewWindow%(19) = 320  '- more width stuff
NewWindow%(20) = 200
NewWindow%(21) = 320
NewWindow%(22) = 200
NewWindow%(23) = &HF  '- Custom screen

' Create a borderless window..could use other types with
' various gadgets if you need them.  For this example,
' we shall not worry about them.

POKEL VARPTR(NewWindow%(7)),&H1800  '- Borderless|Activate

'- get open the screen and window -

' Note that we're using ROM Kernel calls to get these guys
' up, not the usual BASIC WINDOW/SCREEN statements.  This
' means that the runtime system (or interpreter
' environment) doesn't know about them and can't do any
' kind of WINDOW/SCREEN management on them.

OurScreen& = OpenScreen&(VARPTR(NewScreen%(0)))
IF OurScreen& = 0 THEN CALL abort("error during open screen.")

POKEL VARPTR(NewWindow%(15)),OurScreen&
OurWindow&  =  OpenWindow&(VARPTR(NewWindow%(0)))
IF OurWindow& = 0 THEN CALL abort("error during open window.")

'- get pointers to the viewport and rastport -

OurViewPort&  =  ViewPortAddress&(OurWindow&)
OurRastPort&  =  PEEKL(OurWindow& + 50)

'-------------------------
'- Display Heading Text on the Screen                -
'-------------------------

CALL  Move&(OurRastPort&,1,8)
CALL Text&(OurRastPort&,SADD("An Amiga Basic HAM
Screen..."),28)

'-------------------------
'- Use the information we got to draw some lines     -
'- on the HAM screen.                                -
'-                                                   -
'- Note that normal BASIC graphics commands are not  -
'- appropriate.  You must use ROM service routines.  -
'-------------------------

'RANDOMIZE TIMER        '- Used in original version only
starty% = 10
FOR startx% = 1 TO 199
    y% = starty%
    pen% = 0                    '- Start with color reg 0
    FOR x% = startx% TO 320
        pen% = pen% + 1 AND 63  '- Cycle through 0 - 63
         CALL SetAPen(OurRastPort&,pen%)
         CALL WritePixel(OurRastPort&,x%,y%)
    NEXT x%
    'pen% = 0                   '- Reset color reg to 0
    FOR y% = starty% TO 200
        x% = startx% + 1
        pen% = pen% + 1 AND 63  '- Cycle through 0 - 63
        CALL SetAPen(OurRastPort&,pen%)
         CALL WritePixel(OurRastPort&,x%,y%)
    NEXT y%
    starty% = starty% + 1
NEXT startx%

'- All done, pause for about 15 seconds and then exit

a& = TIMER
WHILE TIMER-a& < 15 : WEND

'-------------------------
'- Game Over.  Wrap up everything                    -
'-------------------------

CALL  CleanUpEverything
END

SUB abort(errormessage$)  STATIC
SAY  TRANSLATE$(errormessage$)
CALL  CleanUpEverything
SYSTEM
END SUB

SUB CleanUpEverything  STATIC
SHARED  OurScreen&,OurWindow&
IF OurWindow& <> 0 THEN CALL CloseWindow(OurWindow&)
IF OurScreen& <> 0 THEN CALL CloseScreen(OurScreen&)
LIBRARY CLOSE
END SUB
```

•AC•

ROOMERS

The Bandito

[The statements and projections presented in "Roomers" are rumors in the purest sense. The bits of information are gathered by a third party source from whispers inside the industry. At press time, they remain unconfirmed and are printed for entertainment value only. Accordingly, the staff and associates of Amazing Computing™ *cannot be held responsible for the reports made in this column.]*

The Bandito has told you about HAM Paint Wars, but that's not the only battle raging in the Amiga software market. As more Amigas are sold, the potential software profit gets bigger, and the fight gets rougher. The good old days of homespun software entrepreneurship are almost over.

The word processing market, once a sleepy place first homesteaded by TextCraft and Scribble, has become a crowded battlefield. WordPerfect became the new leader upon introduction, but other contenders are now fighting hard. The latest warrior is excellence! from Micro-Systems Software, an impressive package with a long list of features and a straightforward interface. (The Bandito just wants to know what the slice of pie in their ads has to do with word processing.)

New Horizons is pushing their new version of ProWrite through a broad advertising campaign, but a hard core of Scribble loyalists from the early Amiga days still exists. Meanwhile, WordPerfect Corporation is working on porting WordPerfect 5.0 with full WYSIWYG graphics to the Amiga. Don't expect it any time soon—mid 1989 at best. The MS-DOS version has had many problems, but it is powerful. The Amiga version should be the fully graphic word-processor Amigans expect.

What about TextCraft, the first Amiga word processor? The Bandito heard an interesting tale about that—we'll call it "The Word Processor That Refused To Die." As you may remember, TextCraft was created by Arktronics and distributed by Commodore as one of the first two pieces of Amiga software. Unfortunately, it was wretched—version 1.1 was barely good enough to write an occasional memo, but nothing more. Scribble quickly filled the gap, and TextCraft deservedly faded into oblivion.

The developers, however, thought they had a good thing. Though the original programmer left for another company, Arktronics still had the code. They shopped the product around to the Amiga community, but didn't see much interest. Finally, they tried a different tactic: sell it, but don't tell customers they're buying "used code." Sort of the programming equivalent to rolling back the odometer on a used car. Anyway, Arktronics signed a deal with Electronic Arts to produce a word processor called DeluxeWrite. Of course, they swore up and down that this word processor had nothing to do with TextCraft. It was "all-new code."

Well, the project made it all the way to EA's dealer price list, then things started going wrong. Seems that DeluxeWrite had more bugs than the American embassy in Moscow, was as slow as Jack Tramiel reaching to pick up the check for lunch, and the worst sin of all, wasn't on schedule.

Their suspicions aroused, Electronic Arts performed a close inspection of the code that revealed the original programmer's name still in the remarks. DeluxeWrite was TextCraft with a facelift. So Electronic Arts finally killed DeluxeWrite. Arktronics, now incorporated under a different name (perhaps so people wouldn't know they had done TextCraft), put more work into the program and marketed the program themselves.

Electronic Arts has successfully trained yet another marketing vice-president, but once again, that position is at another company. Karen Janowski, former Director of Marketing for EA Entertainment, is now Vice-President of Marketing at Epyx. Also of interest: Joe Ybarra, an EA founder and VP of roleplaying games at EA, left earlier this year and is now president of Infocom (a division of Mediagenic).

None of these personnel moves seem to have hurt sales any. EA is still the number one "home software" company, and should do about $60 million dollars in sales this year, easily beating out Mediagenic and Broderbund, the nearest competition. Expect some more corpo-

(continued)

rate headhunting among the Big Five—EA, Mediagenic, Broderbund, Mindscape, and Epyx—as they continue to jockey for position.

Mediagenic is still pushing hard. They're finally making a profit after 4 years of losses, but the profit is razor-thin. A couple hundred thou on $10 million every quarter isn't much to write home about. They've made a number of acquisitions to help their position, including the purchase of one of the leading MS-DOS paint program companies, Z-Soft, to secure a position in that market.

Oh, and they're still trying the name-changing strategy. Since the name "Mediagenic" has received near-universal derision (it sounds like a disease that infects television sets), they have now decided that Mediagenic is the name for the corporate holding company that owns Activision, Infocom, Gamestar, and their productivity company that publishes all the HyperCard programs. The name of this company? They're calling it—better sit down—TENpointO. (Please note distinctive spelling.) The Bandito figures they went through that many versions of a real name, then gave up.

While we're on the subject, EA seems to be the only one of the Big Five interested in doing more Amiga software. (Broderbund's release of FantaVision aside.) While they haven't produced much entertainment software for the Amiga lately—with the exception of the excellent Interceptor—work on the productivity software continues.

The Bandito's informers in Sacramento report that DeluxeVideo II should arrive early next year. It will finally support all the different graphics modes, and substantially improve the power and flexibility of presentation.

Caligari, the 3D object creation package, is also moving faster now that Octree has found some programming talent. Caligari should appear around the same time as DVideo II. Unfortunately, it looks like DeluxeProductions won't be upgraded any time soon, but Associated Computer Services has a font creator/titler package that should arrive this fall from EA. The word is that you'll be able to digitize fonts to bring them into this package, which will make font creation (or theft) easier than ever.

Programming talent for the Amiga seems to be in demand. Several Amiga developers are getting serious about finding programmers, going as far as advertising for them in magazines. 68000 assembler programmers can certainly find work if willing to move around the country. Many 6502 assembly language programmers are making the switch to 68000, sensing the imminent demise of the 6502 as a commercially viable software market. Go for it, hackers! There's plenty of untapped power left in the Amiga, and the Bandito is tired of seeing plain old C programs crawling like narcotized slugs across his monitor.

While we're talking software, the Bandito wishes more work would go into interfaces. Some Amiga software today sports dreadful "user-hostile" features. In particular, music software is about as easy to learn as the concert violin.

Some of the worst culprits are direct ports from other machines that don't understand simple Amiga facts such as the two-button mouse...or even that there *is* a mouse). Fortunately, Perry Kivolowitz and Eric Lavitsky of ASDG, with Commodore's backing, formed the Amiga Working Group to deal with such issues and to help advance the Amiga hardware technology. All findings will be put in the public domain.

The Bandito hopes some solid user interface standards that will make the Amiga easier to use will come out of all this. And please, programmers, stay away from color combinations that look like Walt Disney threw up on the screen. Or at least let the user change the colors.

What's happening at Aegis? Bill Volk has gone to Mediagenic to become head of product development for the Activision division (games, that is). John Skeel, former V.P. of marketing, has moved along to Electronic Arts, and Dave Barrett, former Aegis president, has left the nest for unknown pastures. The original Aegis team is now non-existent. Further rumor has it that the marketing staff fled en masse and is now looking for work elsewhere. It also seems as though Aegis's attempt to enter the Mac market has collapsed; the vultures are circling, and buyout rumors are hot and heavy.

Don't fear for the products, though. The Bandito knows they'll find good homes. Plenty of solid products have gathered a good following, and they have a future. Aegis may well pull through this bad time on its own, since they've trimmed back and are now a lean, mean fighting machine. They still have some solid financial backers and, don't forget, several other Amiga companies have pulled through dark times to come back into daylight.

Byte-By-Byte is an example of a company that just won't quit. Scott Peterson has had just about everything that could go wrong happen—remember all those long gone hardware products? But he stuck to his guns, and Sculpt 3D and Animate 3D have proven to be the winners that helped them through. Byte by Byte had an impressive three booths at AmiExpo, and they're still doing Amiga products.

Scott still likes the Amiga and will keep developing for it, despite some nasty business by Commodore. Rumor has it that one Commodore marketing official has been going to user groups and dealers in California telling them not to support Byte-By-Byte because it's a threat to Amiga (rather absurd). In their latest petty move, Commodore put Impulse's Silver in their Siggraph booth instead of Sculpt which is odd because they've always featured Sculpt.

NewTek's new Demo Reel II didn't make it to Ami-Expo, despite a heroic effort. The Bandito's informants report that it was shown to thunderous applause in Palo Alto at the First Amiga User's Group in August. The demo has full-screen, full motion video with sound, including several clips from popular movies, and of course, starring the ever-popular Maxine Headroom. The word overheard: Demo Reel II should appear in September. The Bandito wants to know where the product is, and what exactly it does.

Battlechess, coming up from Interplay Productions, will go after Chessmaster 2000 in the lucrative chess program market. According to Interplay, Battlechess beats Chessmaster in head-to-head competition. The animation of fantasy-like chesspieces battling to the death and digitized sound effects displayed at a recent trade show were superb.

The price increase the Bandito predicted has hit Amigas, but it's too soon to determine the sales effect. Then again, prices are holding steady or rising. They probably won't fall till early next year. Look for an after-Xmas promotion from Commodore to clear out inventory.

The SPA reports micro software sales of $1.82 billion, up 50% from the first half of the year. MS-DOS software sales accounted for about $1.4 billion, Macintosh software about $220 million, Apple II about $37 million. The remaining $160 million is divided up between C64 ($80 million), Amiga, and Atari ST. Doesn't leave much for Amiga, does it? But the Apple II and Atari shares are shrinking, and the Amiga's is growing.

AmiExpo Report

Well, you've heard most of it from everyone else, so the Bandito doesn't need to repeat. A few tidbits stuck out, though, and a few juicy bytes of information leaked out from between the booths. RGB Video, creators of the Deluxe-Help series, had an edit controller in their booth—looks like they're going to become one of the players in the video market.

A-Squared showed off Amiga Live! for the A500 and A2000, meaning they must have come to some sort of deal with Commodore over the marketing rights. Some joker had a 1.2 gigabyte hard drive for the Amiga—just what the Bandito needs.

Wandering around the floor was a copy of Dragon's Lair for the Amiga on six disks! Why so huge? It looks just like the original laser disc arcade game, that's why. Gold Disk was showing Professional Draw, their upcoming object-oriented draw package. It looked good.

Finally, the Bandito notes with some amusement that a popular guessing game among Amiga developers is called "Who's the Bandito?" The players try to figure out the Bandito's identity from the information (and misinformation) in the "Roomers" column. Speculation ranges from folks at Commodore to employees at several Amiga developers. The Bandito is trickier than you think, and his sources are everywhere. In fact, many of the Bandito's best sources don't even know they are his sources. The Bandito is not who you think he is...or is the Bandito a she? Keep guessing, sports fans. •AC•

Amazing Computing's *NEW* Reader Service Card

Wow, AC has done it again!
Due to the tremendous response of our original Reader Service Program, Amazing Computing announces an easier way to contact AC advertisers: The AC Reader Service Card Mailer.

To use the AC system, simply locate all the Amazing Advertisers you wish to contact below, find their appropriate AC Reader Service Card Numbers, and mark them on the card to the right. Fill in your address and mail the card. It is that easy!

The AC Difference
OK, this is not really a new idea. But, we have added a new twist. Gone are the awkward lines "Reader Service Card Number XXX" that reside somewhere on an advertisement. Instead, AC has listed all the advertisers in the Index Of Advertisers with their page numbers and special codes.

Now it is easy to find all the information you want for any AC advertiser in just one place. As always, Amazing Computing has placed their readers first!

Index of Advertisers

COMPUTER AIDED INSTRUCTION (CAI)

A Generalized Authoring System in AmigaBASIC— Part II

[AC V3.9 featured this CAI authoring system written in AmigaBASIC, but we had only enough space to print the TUTOR program. This month we complete the system with the EDITOR Program. —Ed.]

Listing One: EDITOR

```
Editor.VER1.5:

  GOSUB Initialize

Main.1:

  GOSUB Title.Page

    MENU 1,1,1
    MENU 1,2,1
    MENU 1,3,0
    MENU 1,4,0
    MENU 1,5,0
    MENU 1,6,0
    MENU 1,7,0
    MENU 1,8,1
    MENU 1,9,1

  Pick.Loop:
    IF MENU(0)=1 THEN What.You.Want
  GOTO Pick.Loop

  Edit.Old.Lesson:

      ON ERROR GOTO ETrap.One
      CLS
      LOCATE 2,1
      msg$="The following lessons are on this disk"
      PRINT TAB(FNT(80)) msg$
      PRINT
      FILES "This Disk/"

      PRINT TAB(22) "Enter existing or new filename: "

      YText%=CSRLIN-2
      XText%=53
      MaxChar%=15
      MENU 1,8,1
      MENU 1,9,1
      Menu.Chosen%=0
      GOSUB User.Input
      MENU 1,8,0
      MENU 1,9,0
      IF Menu.Chosen%=1 THEN
        ON ERROR GOTO 0
        IF MENU(1)=8 THEN
          GOTO Load.CAI
        ELSEIF MENU(1)=9 THEN
          GOTO Fast.Quit
        END IF
      END IF

      Lesson.Name$=MID$(Fetch$,1,CPoint-1)
      LOCATE YText%+1,1
      msg$="          ..... Looking for "
      msg$=msg$+Lesson.Name$
      msg$=msg$+" ........          "
      PRINT TAB(FNT(80)) msg$

      'if this doesn't work ETrap.One knows file does not exist
      'and asks if you want to create it
      OPEN "This Disk/"+Lesson.Name$ FOR INPUT AS#2
      CLOSE #2
      OPEN "Lessons/"+Lesson.Name$ AS #1 LEN=390
      FIELD #1, 210 AS Q.Field$, 180 AS A.Field$
      OPEN "This Disk/"+Lesson.Name$ AS #2 LEN=2
      FIELD #2, 2 AS How.Many$
      GET #2, 1
      Number.of.Records=CVI(How.Many$)
      Record.Number=Number.of.Records
      ON ERROR GOTO 0

    GOTO Top.Editor


  Delete.Old.Lesson:

      MENU 1,7,1
      MENU 1,8,1
      MENU 1,9,1

      ON ERROR GOTO ETrap.Two
      CLS
      LOCATE 2,20
      PRINT "The following lessons are on this disk"
      PRINT
      FILES "This Disk/"

      PRINT TAB(23) "Which one do you want to delete: "

      YText%=CSRLIN-2
      XText%=56
      MaxChar%=15
      GOSUB User.Input
      MENU 1,7,0
      MENU 1,8,0
      MENU 1,9,0
      IF Menu.Chosen%=1 THEN
        ON ERROR GOTO 0
        IF MENU(1)=7 THEN
          GOTO Main.1
        ELSEIF MENU(1)=8 THEN
          GOTO Load.CAI
        ELSEIF MENU(1)=9 THEN
          GOTO Fast.Quit
        END IF
      END IF

      Delete.Name$=MID$(Fetch$,1,CPoint-1)

      'if this doesn't work ETrap.Two knows file does not exist

      OPEN "This Disk/"+Delete.Name$ FOR INPUT AS#2
      CLOSE #2
      COLOR 3,0
      LOCATE YText%+1,1
        msg$="          ..... Deleting "
        +Delete.Name$+" ........          "
      PRINT TAB(FNT(80)) msg$
      COLOR 1,0
      KILL "Lessons/"+Delete.Name$
      KILL "This Disk/"+Delete.Name$
      ON ERROR GOTO 0

    GOTO Main.1
```

(continued)

```
  Create.New.Lesson:

     'ETrap.One sent you here - to CREATE a new lesson
      CLOSE #2
      OPEN "Lessons/"+Lesson.Name$ AS #1 LEN=390
      FIELD #1, 210 AS Q.Field$, 180 AS A.Field$

      OPEN "This Disk/"+Lesson.Name$ AS #2 LEN=2
      FIELD #2, 2 AS How.Many$

      Number.of.Records=0
      Record.Number=0
      ON ERROR GOTO 0

  GOTO Top.Editor

Top.Editor:

  IF INKEY$<>"" THEN GOTO Top.Editor
    MENU 1,3,1
    MENU 1,4,1
    MENU 1,5,1
    MENU 1,6,1
    MENU 1,7,0

  CLS
  CALL SetFont&(WINDOW(8),garnet16&)
  CALL Move&(WINDOW(8),255,16)
  PRINT "CAI EDITOR"
  CALL SetFont&(WINDOW(8),topaz8&)
  LOCATE 4,4
  PRINT "Editing: ";Lesson.Name$
  LOCATE 4,58
  PRINT "Total Questions =";Number.of.Records
  CALL Move&(WINDOW(8),172,102)
  msg$="Enter <A>dd <D>elete <R>ead <Q>uit"
  CALL Text&(WINDOW(8),SADD(msg$),LEN(msg$))
  COLOR 3,0
  CALL Move&(WINDOW(8),228,102)
  CALL Text&(WINDOW(8),SADD("A"),1)
  CALL Move&(WINDOW(8),276,102)
  CALL Text&(WINDOW(8),SADD("D"),1)
  CALL Move&(WINDOW(8),348,102)
  CALL Text&(WINDOW(8),SADD("R"),1)
  CALL Move&(WINDOW(8),404,102)
  CALL Text&(WINDOW(8),SADD("Q"),1)
  COLOR 1,0
  GOSUB Menu.Prompt
   RWQ.Loop:
      a$=INKEY$
      IF MENU(0)=1 THEN What.You.Want
      IF a$="" THEN RWQ.Loop
      IF UCASE$(a$)="D" THEN GOTO Delete.Record
      IF UCASE$(a$)="R" THEN GOTO Read.Record
      IF UCASE$(a$)="A" THEN GOTO Write.Record
      IF UCASE$(a$)="Q" THEN GOTO Quit.Lesson
    GOTO RWQ.Loop

  Delete.Record:

    MENU 1,3,0
    MENU 1,4,0
    MENU 1,5,0
    MENU 1,6,0
    MENU 1,7,1
    GOSUB Menu.Prompt
    CALL Move&(WINDOW(8),1,102)
    PRINT SPC (19);
        "      Enter question # to delete:              "
    YText%=CSRLIN-2
    XText%=53
    MaxChar%=3
    Menu.Chosen%=0
    GOSUB User.Input
    MENU 1,7,0
    IF Menu.Chosen%=1 THEN
      IF MENU(1)=7 THEN
        GOTO Top.Editor
      END IF
    END IF

    Delete.Record.Number=VAL(MID$(Fetch$,1,CPoint-1))

    IF Delete.Record.Number>Number.of.Records THEN
      BEEP
      CALL Move&(WINDOW(8),1,102)
      msg$="       I can't DELETE question"
      msg$=msg$+STR$(Delete.Record.Number)+" yet.       "
      PRINT TAB(FNT(80)) msg$
      CALL Move&(WINDOW(8),1,114)
        msg$="You have only"+STR$(Number.of.Records)+
        " questions so far"
      PRINT TAB(FNT(80)) msg$
      GOSUB Stop.to.Read
      GOTO Delete.Record
    ELSEIF Delete.Record.Number=0 THEN
      BEEP
      CALL Move&(WINDOW(8),1,90)
      COLOR 3,0
      msg$="       Sorry... invalid question number!       "
      PRINT TAB(FNT(80)) msg$
      GOSUB Stop.to.Read
      COLOR 1,0
      CALL Move&(WINDOW(8),1,90)
      GOSUB Clear.Line
      GOTO Delete.Record
    END IF

    GOSUB Blank.Prompt

    IF Delete.Record.Number<Number.of.Records THEN
      CALL Move&(WINDOW(8),1,102)
      msg$="          ..... Deleting Question"
        +STR$(Delete.Record.Number)
      msg$=msg$+" please wait .....          "
      PRINT TAB(FNT(80)) msg$
        FOR i%=Delete.Record.Number+1 TO Number.of.Records
          GET #1, i%
          PUT #1, i%-1
        NEXT i%
    END IF
      Number.of.Records=Number.of.Records-1
      CALL Move&(WINDOW(8),1,114)
      msg$="          Question"+STR$(Delete.Record.Number)
      msg$=msg$+" has been deleted          "
      PRINT TAB(FNT(80)) msg$
      FOR i%=1 TO 1000:NEXT i%
      GOTO Top.Editor

  Read.Record:

      MENU 1,3,0
      MENU 1,4,0
      MENU 1,5,0
      MENU 1,6,0
      MENU 1,7,1

      GOSUB Menu.Prompt
      CALL Move&(WINDOW(8),1,102)
      msg$="Which question do you want to read:       "
      PRINT SPC(20) msg$

      YText%=CSRLIN-2
      XText%=56
      MaxChar%=3
    Menu.Chosen%=0
    GOSUB User.Input

    MENU 1,7,0
    IF Menu.Chosen%=1 THEN
      IF MENU(1)=7 THEN
        GOTO Top.Editor
      END IF
    END IF

    Record.Number=VAL(MID$(Fetch$,1,CPoint-1))

    IF Record.Number>Number.of.Records THEN
      BEEP
      CALL Move&(WINDOW(8),1,102)
      msg$="     I can't read question"
```

```
    msg$=msg$+STR$(Record.Number)+" yet      "
    PRINT TAB(FNT(80)) msg$
    CALL Move&(WINDOW(8),1,114)
    msg$="    You have only"
      +STR$(Number.of.Records)+" questions so far!    "
    PRINT TAB(FNT(80)) msg$
    GOSUB Stop.to.Read
    GOTO Read.Record
  ELSEIF Record.Number=0 THEN
    BEEP
    CALL Move&(WINDOW(8),1,90)
    COLOR 3,0
    msg$="       Sorry... invalid question number!       "
    PRINT TAB(FNT(80)) msg$
    GOSUB Stop.to.Read
    COLOR 1,0
    CALL Move&(WINDOW(8),1,90)
    GOSUB Clear.Line
    GOTO Read.Record
  END IF

  LOCATE 6,1
  msg$="Reading Question"+STR$(Record.Number)
  PRINT TAB(FNT(80)) msg$

  GOSUB Blank.Prompt

  GET #1, Record.Number

  LINE (30,53)-(610,83), ,b
  LOCATE 8,6
  CALL Text&(WINDOW(8), SADD(Q.Field$), 70)
  LOCATE 9,6
  CALL Text&(WINDOW(8), SADD(Q.Field$)+70, 70)
  LOCATE 10,6
  CALL Text&(WINDOW(8), SADD(Q.Field$)+140, 70)

  LINE (70,101)-(569,130),,b
  LOCATE 14,11
  CALL Text&(WINDOW(8), SADD(A.Field$), 60)
  LOCATE 15,11
  CALL Text&(WINDOW(8), SADD(A.Field$)+60, 60)
  LOCATE 16,11
  CALL Text&(WINDOW(8), SADD(A.Field$)+120, 60)
  GOSUB Stop.to.Read

GOTO Top.Editor

Write.Record:

  MENU 1,3,0
  MENU 1,4,0
  MENU 1,5,0
  MENU 1,6,0
  MENU 1,7,1
  GOSUB Menu.Prompt
  XText%=57
  MaxChar%=3
  Menu.Chosen%=0
  CALL Move&(WINDOW(8),1,102)
  PRINT SPC (21); "Enter question # to add or replace:
   "
  YText%=CSRLIN-2
  GOSUB User.Input
  MENU 1,7,0
  IF Menu.Chosen%=1 AND MENU(1)=7 THEN
    GOTO Top.Editor
  END IF

  Record.Number=VAL(MID$(Fetch$,1,CPoint-1))

  IF Record.Number>Number.of.Records+1 THEN
    BEEP
    CALL Move&(WINDOW(8),1,102)
    msg$="       I can't add question"
    msg$=msg$+STR$(Record.Number)+" yet.       "
    PRINT TAB(FNT(80)) msg$
    CALL Move&(WINDOW(8),1,114)
    msg$="You have only"+STR$(Number.of.Records)
      +" questions so far"
    PRINT TAB(FNT(80)) msg$
    GOSUB Stop.to.Read
    GOTO Write.Record
  ELSEIF Record.Number=0 THEN
    BEEP
    CALL Move&(WINDOW(8),1,90)
    COLOR 3,0
    msg$="       Sorry... invalid question number!       "
    PRINT TAB(FNT(80)) msg$
    GOSUB Stop.to.Read
    COLOR 1,0
    CALL Move&(WINDOW(8),1,90)
    GOSUB Clear.Line
    GOTO Write.Record
  ELSEIF Record.Number=<Number.of.Records THEN
    GET #1, Record.Number
  END IF

  GOSUB Blank.Prompt

  MENU 1,7,1
  GOSUB Input.Question
  MENU 1,7,0

  IF Mind.Change%=1 THEN Top.Editor

  IF Record.Number=Number.of.Records+1 OR UCASE$(r$)="Y" THEN
    LSET Q.Field$=Fetch$
  END IF
  MENU 1,7,1
  GOSUB Input.Answer
  MENU 1,7,0

  IF Mind.Change%=1 THEN Top.Editor

  IF Record.Number=Number.of.Records+1 OR UCASE$(r$)="Y" THEN
    LSET A.Field$=Fetch$
  END IF

  LOCATE 20,30
  COLOR 3,0
  PRINT "Saving question"; Record.Number;
  COLOR 1,0

  PUT #1, Record.Number
  IF Record.Number=Number.of.Records+1 THEN
      Number.of.Records=Number.of.Records+1

  LSET How.Many$=MKI$(Record.Number)
  PUT #2, 1

GOTO Top.Editor:

Input.Question:
  Mind.Change%=0
  MaxChar%=QWidth%*QHeight%
  FOR i%=1 TO QWidth%*3+1
    XCur%(i%)=XQ.Point%(i%)
    YCur%(i%)=YQ.Point%(i%)
  NEXT i%
  XText%=5
  YText%=7
  Wrap1=71
  Wrap2=141

  LINE (30,53)-(610,83),,b
  LOCATE 6,1
  msg$="Enter question"+STR$(Record.Number)
  PRINT TAB(FNT(80)) msg$
  LOCATE 12,1
  IF Record.Number<=Number.of.Records THEN
    msg$="Do you want to change previous question"
    msg$=msg$+STR$(Record.Number)
    msg$=msg$+"?  Yes or No"
    PRINT TAB(FNT(80)) msg$
      LOCATE 8,6
      CALL Text&(WINDOW(8), SADD(Q.Field$), 70)
      LOCATE 9,6
      CALL Text&(WINDOW(8), SADD(Q.Field$)+70, 70)
```

(continued)

```
      LOCATE 10,6
      CALL Text&(WINDOW(8), SADD(Q.Field$)+140, 70)
      GOSUB Clear.Keyboard
      Old.Question:
        r$=INKEY$
        IF UCASE$(r$)="Y" THEN
          LOCATE 12,1
          msg$="        Enter question"+STR$(Record.Number)
          msg$=msg$+" or use PULL DOWN MENU to cancel       "
          PRINT TAB(FNT(80)) msg$
          LOCATE 8,6
          CALL Text&(WINDOW(8),SADD(Empty$),70)
          LOCATE 9,6
          CALL Text&(WINDOW(8),SADD(Empty$),70)
          LOCATE 10,6
          CALL Text&(WINDOW(8),SADD(Empty$),70)
          GOTO Do.It.Question
        ELSEIF UCASE$(r$)="N" THEN
            GOTO Quest.Change.Mind
        ELSEIF MENU(0)=1 AND MENU(1)=7 THEN
            Mind.Change%=1
            GOTO Quest.Change.Mind
        END IF
        GOTO Old.Question
    ELSE
        msg$="        Enter question"+STR$(Record.Number)
        msg$=msg$+" or use PULL DOWN MENU to cancel       "
        PRINT TAB(FNT(80)) msg$

    END IF

    Do.It.Question:
    Menu.Chosen%=0
    GOSUB User.Input

    IF Menu.Chosen%=1 AND MENU(1)=7 THEN
      Mind.Change%=1
    END IF

    Quest.Change.Mind:
    LOCATE 12,4
    GOSUB Clear.Line

RETURN

  Input.Answer:
    Mind.Change%=0
    MaxChar%=AWidth%*AHeight%
    XText%=10
    YText%=13
    FOR i%=1 TO AWidth%*3+1
      XCur%(i%)=XA.Point%(i%)
      YCur%(i%)=YA.Point%(i%)
    NEXT i%
    Wrap1=61
    Wrap2=121
    Menu.Chosen%=0

    LINE (70,101)-(569,130),,b
    LOCATE 18,1
    IF Record.Number<=Number.of.Records THEN
      msg$="Do you want to change previous answer"
      msg$=msg$+STR$(Record.Number)
      msg$=msg$+"?  Yes or No"
      PRINT TAB(FNT(80)) msg$
        LOCATE 14,11
        CALL Text&(WINDOW(8), SADD(A.Field$), 60)
        LOCATE 15,11
        CALL Text&(WINDOW(8), SADD(A.Field$)+60, 60)
        LOCATE 16,11
        CALL Text&(WINDOW(8), SADD(A.Field$)+120, 60)
        GOSUB Clear.Keyboard
        Old.Answer:
          r$=INKEY$
          IF UCASE$(r$)="Y" THEN
            LOCATE 18,1
            msg$="        Enter answer"+STR$(Record.Number)
            msg$=msg$+" or use PULL DOWN MENU to cancel       "
            PRINT TAB(FNT(80)) msg$
            LOCATE 14,11
            CALL Text&(WINDOW(8), SADD(Empty$), 60)
            LOCATE 15,11
            CALL Text&(WINDOW(8), SADD(Empty$)+60, 60)
            LOCATE 16,11
            CALL Text&(WINDOW(8), SADD(Empty$)+120, 60)
            GOTO Do.It.Answer
          ELSEIF UCASE$(r$)="N" THEN
            GOTO Ans.Change.Mind
          ELSEIF MENU(0)=1 AND MENU(1)=7 THEN
              Mind.Change%=1
              GOTO Ans.Change.Mind
          END IF
          GOTO Old.Answer
    ELSE
      LOCATE 18,1
      msg$="        Enter answer"+STR$(Record.Number)
      msg$=msg$+" or use PULL DOWN MENU to cancel       "
      PRINT TAB(FNT(80)) msg$
    END IF

    Do.It.Answer:
    Menu.Chosen%=0
    GOSUB User.Input

    IF Menu.Chosen%=1 AND MENU(1)=7 THEN
      Mind.Change%=1
    END IF

    Ans.Change.Mind:

    LOCATE 18,10
    GOSUB Clear.Line

  RETURN


  User.Input:

    IF INKEY$<>"" THEN GOTO User.Input

    'Initialize CPoint at beginning of text
    CPoint=1

    Fetch$=Empty$

    Very.Top.of.Loop:
      'Initialize Cursor
      Cur$="_"

      Top.of.Loop:
        'Place cursor at current position
        LOCATE YText%+YCur%(CPoint),XText%+XCur%(CPoint)
        PRINT Cur$

        'Change cursor so that it will blink
        IF Cur$="_" THEN
            Cur$=" "
        ELSEIF Cur$=" " THEN
            Cur$="_"
        END IF

        'Initialize cursor timer
        T=0

        In.Loop:
          a$=INKEY$
          IF MENU(0)=1 THEN
            Menu.Chosen%=1
            GOTO Input.Done
          END IF

          'Test if something was entered
          IF a$<>"" THEN Test

          'Nothing entered....increment cursor timer
          T=T+1
```

```
      'Test if cursor needs blinking
       IF T=50 THEN Top.of.Loop

       GOTO In.Loop

  Test:

   'Test for carriage return
    IF a$=CHR$(13) THEN
      LOCATE YText%+YCur%(CPoint),XText%+XCur%(CPoint)
      PRINT " "
      GOTO Input.Done
    END IF

    'Test for backspace key
    IF a$=CHR$(8) THEN Back.Up

   'Test for maximum number of characters
    IF CPoint=MaxChar%+1 THEN In.Loop

    'Test for illegal character
    IF ASC(a$)<31 AND ASC(a$)>127 THEN In.Loop

    'Test if word wrap is needed
    IF CPoint=Wrap1 OR CPoint=Wrap2 THEN
      x=0
      Count:
        IF MID$(Fetch$,CPoint-x-1,1)<>" " THEN
          x=x+1
          IF x=30 THEN No.Wrap
          GOTO Count
        END IF
      temp$=MID$(Fetch$,CPoint-x,x)

      MID$(Fetch$,CPoint-x,x)="

      MID$(Fetch$,CPoint,x)=temp$

      LOCATE YText%+YCur%(CPoint-x),XText%+XCur%(CPoint-x)
      PRINT MID$(Fetch$,CPoint-x,x)

      LOCATE YText%+YCur%(CPoint),XText%+XCur%(CPoint)
      PRINT MID$(Fetch$,CPoint,x)

      CPoint=CPoint+x

      No.Wrap:
    END IF

  'Input must be legitimate!!!

  'Echo input to screen
  LOCATE YText%+YCur%(CPoint),XText%+XCur%(CPoint)
  PRINT a$

  'Store input in Fetch$ variable
  MID$(Fetch$,CPoint,1)=a$


  'Increment CPoint
  CPoint=CPoint+1

GOTO Very.Top.of.Loop

Back.Up:

  'Test if cursor is at beginning of text
  IF CPoint=1 THEN In.Loop

  'Erase cursor from present position
  LOCATE YText%+YCur%(CPoint),XText%+XCur%(CPoint)
  PRINT " "

  'Decrement CPoint
  CPoint=CPoint-1

  'Erase character from Fetch$ variable
  MID$(Fetch$,CPoint,1)=" "

GOTO Top.of.Loop

  Input.Done:

RETURN

Initialize:

  DEF FNT(Z)=INT((Z-LEN(msg$))/2)
    SCREEN 1,640,200,2,2
    WINDOW 2,,(0,0)-(631,186),16, 1
    LOCATE 10,29
    PRINT "... Please wait ..."

  LIBRARY "graphics.library"
  LIBRARY "diskfont.library"

  DECLARE FUNCTION OpenDiskFont&() LIBRARY
  DECLARE FUNCTION OpenFont&() LIBRARY

  DIM TextAttr&(1)
  TextAttr&(0)=SADD("topaz.font"+CHR$(0))
  TextAttr&(1)=8*65536&
  topaz8&=OpenFont&(VARPTR(TextAttr&(0)))
  IF topaz8&=0 THEN
      PRINT "I can't find topaz 8 font"
      FOR i%=1 TO 1000:NEXT i%
      GOTO Fast.Quit
  END IF

  TextAttr&(0)=SADD("topaz.font"+CHR$(0))
  TextAttr&(1)=11*65536&
  topaz11&=OpenDiskFont&(VARPTR(TextAttr&(0)))
  IF topaz11&=0 THEN
      PRINT "I can't find topaz 11 font"
      FOR i%=1 TO 1000:NEXT i%
      GOTO Fast.Quit
  END IF

  TextAttr&(0)=SADD("garnet.font"+CHR$(0))
  TextAttr&(1)=16*65536&
  garnet16&=OpenDiskFont&(VARPTR(TextAttr&(0)))
  IF topaz11&=0 THEN
      PRINT "I can't find garnet 16 font"
      FOR i%=1 TO 1000:NEXT i%
      GOTO Fast.Quit
  END IF

    MENU 1,0,1,"Things you can do   "
    MENU 1,1,0,"Open/Create Lesson "
    MENU 1,2,0,"DELETE Lesson File "
    MENU 1,3,0,"QUIT Lesson      <Q>"
    MENU 1,4,0,"READ Question    <R>"
    MENU 1,5,0,"ADD   Question   <A>"
    MENU 1,6,0,"DELETE Question <D>"
    MENU 1,7,0,"I changed my mind  "
    MENU 1,8,0,"GOTO Tutor         "
    MENU 1,9,0,"GOTO Workbench     "

    MENU 2,0,0,""
    MENU 3,0,0,""
    MENU 4,0,0,""

    QWidth%=70
    QHeight%=3
    DIM XQ.Point%(QWidth%*3+1)
    DIM YQ.Point%(QWidth%*3+1)
    DIM XCur%(3*QWidth%+1)
    DIM YCur%(3*QWidth%+1)

    AWidth%=60
    AHeight%=3
    DIM XA.Point%(AWidth%*3+1)
    DIM YA.Point%(AWidth%*3+1)

    FOR i%=1 TO QWidth%
      XQ.Point%(i%)=i%
      YQ.Point%(i%)=1
      XQ.Point%(QWidth%+i%)=i%
      YQ.Point%(QWidth%+i%)=2
      XQ.Point%(2*QWidth%+i%)=i%
      YQ.Point%(2*QWidth%+i%)=3
```

(continued)

```
    NEXT i%
    XQ.Point%(3*QWidth%+1)=QWidth%+1
    YQ.Point%(3*QWidth%+1)=3

    FOR i%=1 TO AWidth%
      XA.Point%(i%)=i%
      YA.Point%(i%)=1
      XA.Point%(AWidth%+i%)=i%
      YA.Point%(AWidth%+i%)=2
      XA.Point%(2*AWidth%+i%)=i%
      YA.Point%(2*AWidth%+i%)=3
    NEXT i%
    XA.Point%(3*AWidth%+1)=61
    YA.Point%(3*AWidth%+1)=3

    FOR i%=1 TO 15
      XCur%(i%)=i%
      YCur%(i%)=1
    NEXT i%

    Empty$="           "
    FOR i%=1 TO 24
      Empty$=Empty$+"           "
    NEXT i%

RETURN

Title.Page:
    CLS
    LINE (175,13)-(445,55),1,b
    LINE (174,14)-(174,56),2
    LINE (173,15)-(173,57),2
    LINE -(443,57),2
    LINE (171,15)-(171,57),2
    LINE -(441,57),2

    CALL SetFont&(WINDOW(8),topaz11&)
    CALL Move&(WINDOW(8),1,27)
    msg$="Computer Aided Instruction"
    PRINT TAB(FNT(80)) msg$

    CALL SetFont&(WINDOW(8),garnet16&)
    CALL Move&(WINDOW(8),280,105)
    PRINT "EDITOR"

    CALL SetFont&(WINDOW(8),topaz8&)
    CALL Move&(WINDOW(8),1,47)
    msg$="Revision 1.5"
    PRINT TAB(FNT(80)) msg$
    CALL Move&(WINDOW(8),1,75)
    msg$="by Paul Castonguay"
    PRINT TAB(FNT(80)) msg$

    CALL Move&(WINDOW(8),1,130)
    msg$="This program allows you to ENTER material
        that you wish to"
    PRINT TAB(FNT(80)) msg$
    CALL Move&(WINDOW(8),1,140)
    msg$="study using the CAI.Program in this package."
    PRINT TAB(FNT(80)) msg$

    COLOR 3,0
    CALL Move&(WINDOW(8),1,179)
    msg$="Press RIGHT mouse button and"
    PRINT TAB(FNT(80)) msg$
    CALL Move&(WINDOW(8),1,189)
    msg$="Select from PULL DOWN MENU"
    PRINT TAB(FNT(80)) msg$;
    COLOR 1,0

RETURN

What.You.Want:

      MENU 1,1,0
      MENU 1,2,0
      MENU 1,3,0
      MENU 1,4,0
      MENU 1,5,0
      MENU 1,6,0
      MENU 1,7,0
      MENU 1,8,0
      MENU 1,9,0

  IF MENU(1)=1 THEN
      GOTO Edit.Old.Lesson
  ELSEIF MENU(1)=2 THEN
      GOTO Delete.Old.Lesson
  ELSEIF MENU(1)=3 THEN
      GOTO Quit.Lesson
  ELSEIF MENU(1)=4 THEN
      GOTO Read.Record
  ELSEIF MENU(1)=5 THEN
      GOTO Write.Record
  ELSEIF MENU(1)=6 THEN
      GOTO Delete.Record
  ELSEIF MENU(1)=7 THEN
      GOTO Top.Editor
  ELSEIF MENU (1)=8 THEN
      GOTO Load.CAI
  ELSE
      GOTO Fast.Quit
  END IF


Quit.Editor:

  MENU RESET

  LSET How.Many$=MKI$(Number.of.Records)
  PUT #2, 1

  CLOSE #2
  CLOSE #1

  ON ERROR GOTO 0
  ON ERROR GOTO Erase.Info.Icon.2
  KILL "This Disk/"+Lesson.Name$+".info"
  Info.Icon.Gone.2:
  WINDOW CLOSE 2
  SCREEN CLOSE 1
  END

Erase.Info.Icon.2:
  RESUME Info.Icon.Gone.2

 Fast.Quit:
   MENU RESET
   WINDOW CLOSE 2
   SCREEN CLOSE 1
   END

 Load.CAI:
   MENU RESET
   WINDOW CLOSE 2
   SCREEN CLOSE 1
   RUN "Tutor"

 Quit.Lesson:
   CALL Move&(WINDOW(8),1,114)
   PRINT "
    "
   CALL Move&(WINDOW(8),1,102)
   msg$="    .... Please wait while I save your files ....    "
   PRINT TAB(FNT(80)) msg$
   LSET How.Many$=MKI$(Number.of.Records)
   PUT #2, 1
   CLS
   CLOSE #2
   CLOSE #1

   ON ERROR GOTO 0
   ON ERROR GOTO Erase.Info.Icon.1
   KILL "This Disk/"+Lesson.Name$+".info"
   Info.Icon.Gone.1:
   ON ERROR GOTO 0
   WINDOW 2
   GOTO Main.1
```

```
Erase.Info.Icon.1:
  RESUME Info.Icon.Gone.1

Menu.Prompt:
  CALL Move&(WINDOW(8),1,114)
  msg$="          or use PULL DOWN MENU          "
  PRINT TAB(FNT(80)) msg$
RETURN

Blank.Prompt:
    CALL Move&(WINDOW(8),1,102)
    GOSUB Clear.Line
    CALL Move&(WINDOW(8),1,114)
    GOSUB Clear.Line
RETURN

Clear.Line:
  CALL Text&(WINDOW(8),SADD(Empty$),75)
RETURN

Stop.to.Read:
  LOCATE 22,1
  msg$="Press LEFT mouse button or <RETURN>"
  PRINT TAB(FNT(80)) msg$
    WHILE MOUSE(0)=0
      IF INKEY$=CHR$(13) THEN Leave
    WEND
    WHILE MOUSE(0)<>0
      IF INKEY$=CHR$(13) THEN Leave
    WEND
    Leave:
      LOCATE 22,1
      GOSUB Clear.Line
RETURN

Clear.Keyboard:
  IF INKEY$<>"" THEN Clear.Keyboard
RETURN

ETrap.One:

  BEEP
  WINDOW 2
  IF ERR=53 THEN
    request1$="There is no "+Lesson.Name$
    request2$="Want to CREATE it?"
    GOTO ExitError1
  END IF

  request1$="ERROR NUMBER"+STR$(ERR)
  request2$=""
  GOTO ExitError2

ETrap.Two:

  BEEP
  WINDOW 2
  IF ERR=53 THEN
    request1$="There is no "+Delete.Name$
    request2$="I cannot DELETE!"
    GOTO ExitError2
  END IF

  request1$="ERROR NUMBER"+STR$(ERR)
  request2$=""
  GOTO ExitError2

  ExitError1:

    'CALL Requester(request1$,request2$,"yes","NO",2,Answer%)
    box1$="yes"
    box2$="NO"
    default%=2
    GOSUB AlertBox
    IF Answer%=1 THEN
      CLOSE #2
      RESUME Create.New.Lesson
    ELSE
      CLOSE #2
      RESUME Edit.Old.Lesson
    END IF
```



```
  ExitError2:

    'CALL Requester(request1$,request2$,"Try
Again","Workbench",2,Answer%)
    box1$="Try Again"
    box2$="Workbench"
    default%=2
    GOSUB AlertBox
    IF Answer%=1 THEN
      CLOSE #2
      RESUME Delete.Old.Lesson
    ELSE
      CLOSE #2
      RESUME Fast.Quit
    END IF

AlertBox:

  WINDOW 3,"Program Request",(0,0)-(311,45),16,1
  PRINT LEFT$(request1$,39)
  PRINT LEFT$(request2$,39)
  b1$=LEFT$(box1$,12)
  b2$=LEFT$(box2$,12)
  boxsize1=(LEN(b1$)+2)*10
  boxsize2=(LEN(b2$)+2)*10
  x1=(312-(boxsize1+boxsize2))/3
  x2=x1+boxsize1
  x3=x1+x2
  x4=x3+boxsize2
  LINE(x1,20)-(x2,38),2,b
  LINE(x3,20)-(x4,38),2,b
  IF default%=1 THEN LINE(x1+2,22)-(x2-2,36),3,b
  IF default%=2 THEN LINE(x3+2,22)-(x4-2,36),3,b
  LOCATE 4,1
  PRINT PTAB(x1+10);b1$;
  PRINT PTAB(x3+10);b2$

  Reqloop:

    WHILE MOUSE(0)=0:WEND

    m1=MOUSE(1)
    m2=MOUSE(2)

    IF m1>x1 AND m1<x2 AND m2>20 AND m2<38 THEN
      Answer%=1
      LINE(x1,20)-(x2,38),1,bf
    ELSEIF m1>x3 AND m1<x4 AND m2>20 AND m2<38 THEN
      Answer%=0
      LINE(x3,20)-(x4,38),1,bf
    ELSE
      GOTO Reqloop
    END IF

  WHILE MOUSE(0)<>0:WEND
  WINDOW CLOSE 3
RETURN
```

•AC•

PD Serendipity

Insight into the World of Freely Redistributable Software for the Amiga.

by C.W. Flatte

Hey there! As f I wasn't already behind on my reporting of the latest Fred Fish disks...I turn my head for a second, and POOF! Fred Fish 147-154. This month I'm extra tight on space and time, so I'll try to briefly cover as much as I can. As always, check out the PDS catalog on page 105 for the complete scoop.

Fred Fish 147
MicroGNUEmacs (MG 2b)
Always getting better, MicroGNUemacs is now in version MG 2b. There have been many additions and enhancements since the original version of this timeless text editor by Dave Conroy. In order to fit all the files on one disk and preserve the original Workbench environment, the source code files have been archived with Zoo. (A copy of Zoo is provided.)

Fred Fish 148
EFJ
"Escape from Jovi" is a fast-moving, action-packed game featuring hi-res scrolling, a large playfield, disk-based hi-score list, stereo sound, and multiple levels. Includes only the executable. EFJ is shareware by Oliver Wagner.

Fme
This one is for Fire Power addicts. This well-done map editor for the FirePower game features interlaced hi-res with an intuition interface. Fme also includes instructions on how to make Fme a bootable disk. Fme is by Gregory MacKay and includes the source.

HandyIcons
This really neat tool adds a menustrip to the Workbench window that allows you to run selected Workbench tools by menu selection. The program can be set up to provide custom environments. Note, however, that the current version supports Workbench tools, but not WB projects. HandyIcon includes only the binary and was created by Alan Rubright.

Scrambler v0.01
Are you hiding something...or do need to? Keep that text file from prying eyes with Scrambler by Foster Hall, a simple program that encodes/decodes a text file into illegible gibberish that resembles executable code.

Fred Fish149
AnimalSounds
Moooo...Oink-Oink. AnimalSounds is a collection of digitized animal sounds by The Trumor Company, Inc. Also included is a simple sound player by Don Pitts.

DX-VoiceSorter
Attention DX-Synthesizer fans... DX-VoiceSorter, to be used with Jack Deckard's VoiceFiler (Fred Fish 82), sorts any number of voicefiles. Using VoiceFiler, the files are then stored in a new voicefile of bits from various files. This one includes the source and is by Dave Boucher.

Keep V 1.2
Are you an online junkie? Do you cringe when you look at your phone bill? Keep is a friendly utility for BBS and network junkies who download messages in one large file and then read them off-line. Using only the mouse, you can examine such files one message at a time and tag those you wish to keep. Keep is by Tim Grantham and includes only the executable.

Less V1.3
More or Less? Similar to the Unix program "More," Less, by Mark Nudelman, is even better. The program has forward and backward scrolling, searching and positioning by percent of file and line number, etc. You can now also print the current file. Very useful! Version 1.3 is an update of the version on disk 92. Includes source. Amiga port by Bob Leivian

Scheme
Do you speak with a LISP? Scheme is a statically-scoped and properly tail-recursive dialect of the LISP programming language. Invented by Guy Lewis Steele Jr. and Gerald Jay Sussman. Includes only the binary. Amiga port by Ed Puckett.

Gotta go! Until next time...
Gotcha!

—C.W. Flatte

•AC•

The AMICUS & Fred Fish

Public Domain Software Library

This software is collected from user groups and electronic bulletin boards around the nation. Each Amicus disk is nearly full, and is fully accessible from the Workbench. If source code is provided for any program, then the executable version is also present. This means that you don't need the C compiler to run these programs. An exception is granted for those programs only of use to people who own a C compiler.

The Fred Fish disk are collected by Mr. Fred Fish, a good and active friend of the Amiga.

Note: Each description line below may include something like 'S-O-E-D', which stands for 'source, object file, executable and documentation'. Any combination of these letters indicates what forms of the program are present. Basic programs are presented entirely in source code format.

AMICUS Disk 1

ABasic programs: Graphics

3DSolids	3d solids modeling prog. w/sample data files
Blocks	draws blocks
Cubes	draws cubes
Durer	draws pictures in the style of Durer
FScape	draws fractal landscapes
Hidden	3D drawing program, w/ hidden line removal
JPad	simple paint program
Optical	draw several optical illusions
PaintBox	simple paint program
Shuttle	draws the Shuttle in 3d wireframe
SpaceArt	graphics demo
Speaker	speech utility
Sphere	draws spheres
Spiral	draws color spirals
ThreeDee	3d function plots
Topography	artificial topography
Wheels	draws circle graphics
Xenos	draws fractal planet landscapes

ABasic programs: Tools

AddressBook	simple database program for addresses
CardFile	simple card file database program
Demo	multiwindow demo
KeyCodes	shows keycodes for a key you press
Menu	run many ABasic programs from a menu
MoreColors	way to get more colors on the screen at once, using aliasing
shapes	simple color shape designer SpeakIt speech and narrator demo

ABasic programs: Games

BrickOut	classic computer brick wall game
Othello	also known as 'go'
Saucer	simple shoot-em-up game
Spelling	simple talking spelling game
ToyBox	selectable graphics demo

ABasic programs: Sounds

Entertainer	plays that tune
HAL9000	pretends it's a real computer
Police	simple police siren sound
SugarPlum	plays "The Dance of the Sugarplum Fairies"

C programs:

ATerm	simple terminal program, S-E
cc	aid to compiling with Lattice C
decvnt	opposite of CONVERT for cross developers
Dotty	source code to the 'dotty window demo
echox	unix-style filename expansion, partial S,O-D
fasterfp	explains use of fast-floating point math
FixDate	fixes future dates on all files on a disk, S-E
freedraw	simple Workbench drawing prog.,S-E
GfxMem	graphic memory usage indicator, S-E
Grep	searches for a given string in a file with
docs.	ham shows off the hold-and-modify method of color generation
IBM2Amiga	fast parallel cable transfers between an IBM and an Amiga
Mandel	Mandelbrot set program, S-E
moire	patterned graphic demo, S-E
objfix	makes Lattice C object file symbols visible to Wack, S-E
quick	quick sort strings routine
raw	example sample window I/O
setlace	turns on interlace mode, S-E
sparks	qix-type graphic demo, S-E

Other executable programs:

SpeechToy	speech demonstration
WhichFont	displays all available fonts

Texts:

68020	describes 68020 speedup board from CSA
Aliases	explains uses of the ASSIGN command
Bugs	known bug list in Lattice C 3.02
CLICard	reference card for AmigaDOS CLI
CLICommands	guide to using the CLI
Commands	shorter guide to AmigaDOS CLI commands
EdCommands	guide to the ED editor
Filenames	AmigaDOS filename wildcard conventions
HalfBright	explains rare graphics chips that can do more colors
ModemPins	description of the serial port pinout
RAMdisks	tips on setting up your RAM: disk
ROMWack	tips on using ROMWack
Sounds	explanation of Instrument demo sound file format
Speed	refutation of Amiga's CPU and custom chip speed
WackCmds	tips on using Wack

AMICUS Disk 2

C programs:

alib	AmigaDOS object library manager , S-E
ar	text file archive program, S-E
fixobj	auto-chops executable files
shell	simple CLI shell, S-E
sq, usq	file compression programs, S-E
YachtC	a familiar game, S-E
Make	a simple 'make' programming utility, S-E
Emacs	an early version of the Amiga text editor, S-E-D

Assembler programs:

bsearch.asm	binary search code
qsort.asm	Unix compatible qsort() function, source and C test program
setjmp.asm	setjmp() code for Lattice 3.02
SVprintf	Unix system V compatible printf()
trees.o	Unix compatible tree() function, O-D

(This disk formerly had IFF specification files and examples. Since this spec is constantly updated, the IFF spec files have been moved to their own disk in the AMICUS collection.)

John Draper Amiga Tutorials:

Animate	describes animation algorithms
Gadgets	tutorial on gadgets
Menus	learn about Intuition menus

AMICUS Disk 3

C programs:

Xref	a C cross-reference gen., S-E
6bitcolor	extra-half-bright chip gfx demo, S-E
Chop	truncate (chop) files down to size, S-E
Cleanup	removes strange characters from text files
CR2LF	converts carriage returns to line feeds in Amiga files, S-E
Error	adds compile errors to a C file, S
Hello	window ex. from the RKM, S
Kermit	generic Kermit implementation, flakey, no terminal mode, S-E
Scales	sound demo plays scales, S-E
SkewB	Rubik cube demo in hi-res colors, S-E

AmigaBasicProgs(dir)

Automata	cellular automata simulation
CrazyEights	card game
Graph	function graphing programs
WitchingHour	a game

ABasiC programs:

Casino	games of poker, blackjack, dice, and craps
Gomoku	also known as 'othello'
Sabotage	sort of an adventure game

Executable programs:

Disassem	a 68000 disassembler, E-D
DpSlide	shows a given set of IFF pictures, E-D
Arrange	a text formatting program, E-D

Assembler programs:

Argoterm	terminal program with speech and Xmodem, S-E

AMICUS Disk 4 Files from the original Amiga Technical BBS

Note that some of these files are old, and refer to older versions of the operating system. These files came from the Sun system that served as Amiga technical support HQ for most of 1985. These files do not carry a warranty, and are for educational purposes only. Of course, that's not to say they don't work.

Complete and nearly up-to-date C source to 'image.ed', an early version of the Icon Editor. This is a little flaky, but compiles and runs.

An Intuition demo, in full C source, including files: demomenu.c, demomenu2.c, demoreq.c, getascii.c, idemo.c, idemo.guide, idemo.make, idemoall.h, nodos.c, and txwrite.c

addmem.c	add external memory to the system
bobtest.c	example of BOB use
consoleIO.c	console IO example
creaport.c	create and delete ports
creastdi.c	create standard I/O requests
creatask.c	creating task examples
diskio.c	example of track read and write
dotty.c	source to the 'dotty window' demo
dualplay.c	dual playfield example
flood.c	flood fill example
freemap.c	old version of 'freemap'
geltools.c	tools for VSprites and BOBs
gfxmem.c	graphic memory usage indicator
hello.c	window example from RKM
inputdev.c	adding an input handler to the input stream
joystik.c	reading the joystick
keybd.c	direct keyboard reading
layertes.c	layers examples
mousport.c	test mouse port
ownlib.c, ownlib.asm	example of making your own library with Lattice
paratest.c	tests parallel port commands
seritest.c	tests serial port commands
serisamp.c	example of serial port use
prinintr.c	sample printer interface code
prtbase.h	printer device definitions
regintes.c	region test program
setlace.c	source to interlace on/off program
setparallel.c	set the attributes of the parallel port
SetSerial.c	set the attributes (parity, data bits, baud rate, etc.) of the
singplay.c	single playfield example
speechtoy.c	source to narrator and phonetics demo
timedely.c	simple timer demo
timer.c	exec support timer functions
timrstuf.c	more exec support timer functions
WhichFont.c	loads and displays all available system fonts

process.i and prtbase.i assmebler include files:

autorqstr.txt	warnings of deadlocks with autorequesters
consoleIO.txt	copy of the RKM console I/O chapter
diskfont.txt	warning of disk font loading bug
fullfunc.txt	list of #defines, macros, functions
inputdev.txt	preliminary copy of the input device chapter

License information on Workbench distribution license

printer pre-release copy of the chapter on printer drivers, from RKM 1.1 v11ld.txt 'diff' of .ld file changes from version 1.0 to 1.1 v28v1.diff 'diff' of include file changes from version 28 to 1.0

AMICUS Disk 5 Files from the Amiga Link / Amiga Information Network

Note that some of these files are old, and refer to older versions of the operating system. These files are from Amiga Link. For a time, Commodore supported Amiga Link, aka AIN, for online developer technical support. It was only up and running for several weeks. These files do not carry a warranty, and are for educational purposes only. Of course, that's not to say they don't work.

A demo of Intuition menus called 'menudemo', in C source

whereis.c	find a file searching all subdirectories
bobtest.c	BOB programming example
sweep.c	sound synthesis example

Assembler files:

mydev.asm	sample device driver
mylib.asm	sample library example
mylib.i	
mydev.i	
asmsupp.i	
macros.i	assembler include files

Texts:

amigatricks	tips on CLI commands
extdisk	external disk specification
gameport	game port spec
parallel	parallel port spec
serial	serial port spec
v1.1update	list of new features in version 1.1
v1.1h.txt	'diff' of include file changes from version

Files for building your own printer drivers, including dospecial.c, epsondata.c, init.asm, printer.c, printer.link, printertag.asm, render.c, and wait.asm. This disk does contain a number of files describing the IFF specification. These are not the latest and greatest files, but remain here for historical purposes. They include text files and C source examples. The latest IFF spec is elsewhere in this library.

AMICUS Disk 6 IFF Pictures

This disk includes the DPSlide program, which can view a given series of IFF pictures, and the 'showpic' program, which can view each file at the click of an icon. The pictures include a screen from ArticFox, a Degas dancer, the guys at Electronic Arts, a gorilla, horses, King Tut, a lighthouse, a screen from Marble Madness, the Bugs Bunny Martian, a still from an old movie, the Dire Straits moving company, a screen from Pinball Contruction Set, a TV newcaster, the PaintCan, a world map, a Porsche, a shuttle mission patch, a tyrannosaurus rex, a planet view, a VISA card, and a ten-speed.

AMICUS Disk 7 DigiView HAM demo picture disk

This disk has pictures from the DigiView hold-and-modify video digitizer. It includes the ladies with pencils and lollypops, the young girl, the bulldozer, the horse and buggy, the Byte cover, the dictionary page, the robot and Robert. This includes a program to view each picture separately, and all together as separate, slidable screens. The 'seeilbm' program, to turn any screen into an IFF picture.

AMICUS Disk 8

C programs:

Browse	view text files on a disk, using menus S-E-D
Crunch	removes comments and white space from C files, S-E
IconExec	EXECUTE a series of commands from Workbench S-E
PDScreen Dump	dumps Rastport of highest screen to printer
SetAlternate	sets a second image for an icon, when clicked once S-E
SetWindow	makes windows for a CLI program to run under Workbench S-E
SmallClock	a small digital clockin a window menu bar
Scrimper	the screen printer in the fourth AC S-E

Amiga Basic Programs:

(**Note:** Many of these programs are present on AMICUS Disk 1. Several of these were converted to Amiga Basic, and are included here.)

AddressBook	a simple address book database
Ball	draws a ball
Cload	program to convert Compuserve hex files to binary, S-D
Clue	the game, Intuition driven
ColorArt	art drawing program
DeluxeDraw	the drawing program in the 3rd AC, S-D
Eliza	conversational computer psychologist
Othello	the game, as known as 'go'
RatMaze	3D ratmaze game
ROR	boggling graphics demo
Shuttle	draws 3D pictures of the space shuttle
Spelling	simple spelling program
YoYo	wierd zero-gravity yo-yo demo, tracks yo-yo to the mouse

Executable programs:

3Dcube	Modula-2 demo of a rotating cube
AltIcon	sets a second icon image, displayed when the icon is clicked
AmigaSpell	a slow but simple spell checker, E-D
arc	the ARC file compression progra must-have for telecom, E-D
Bertrand	graphics demo
disksalvage	prog. to rescue trashed disks, E-D
KwikCopy	a quick but nasty disk copy program: ignores errors, E-D
LibDir	lists hunks in an object file E-D
SaveILBM	saves any screen as IFF pic.E-D ??
ScreenDump	shareware screen dump prog, E only
StarTerm	version 2.0, term program, XmodemE-D

Texts:

LatticeMain	tips on fixing _main.c in Lattice
GDiskDrive	make your own 5 1/4 drive
GuruMed	explains the Guru numbers
Lat3.03bugs	bug list of Lattice C version 3.03
MForgeRev	user's view of the MicroForge HD
PrintSpooler	EXECUTE-based print spool prog.

.BMAP files:

These are the necessary links between Amiga Basic and the system libraries. To take advantage of the Amiga's capabilities in Basic, you need these files. BMAPs are included for 'clist', 'console', 'diskfont', 'exec', 'icon', 'intuition', 'layers', 'mathffp', mathieeedoubas', 'mathieeesingbas', 'mathtrans', 'polgo', 'timer' and 'translator'.

AMICUS Disk 9

Amiga Basic Programs:

FlightSim	simple flight simulator program
HuePalette	explains Hue, Saturation, & Intensity
Requester	ex. of requesters from Amiga Basic
ScrollDemo	demonstrates scrolling capabilities
Synthesizer	sound program
WorldMap	draws a map of the world

Executable programs:

Boing!	latest Boing! demo,with selectable speed,E
Brush2C	converts an IFF brush to C data instructions, initialization code, E
Brush2Icon	converts IFF brush to an icon, E
Dazzle	graphics demo, tracks to mouse, E
DeciGEL	assembler program for stopping 68010 errors, S-E-D
Klock	menu-bar clock and date display, E
life	the game of life, E
TimeSet	Intuition-based way to set the time & date
EMEmacs	another Emacs, more oriented to word processing, S-E-D
MyCLI	a CLI shell, works without the Workbench, S-E-D

Texts:

FnctnKeys	read function keys from Amiga Basic
HackerSin	explains how to win the game 'hacker'
Ist68010	guide to installing a 68010 in your Amiga
Boing!	latest Boing! demo,with selectablespeed, E
Brush2C	converts an IFF brush to C data instructions, initialization code, E
Brush2Icon	converts IFF brush to an icon, E
Dazzle	graphics demo, tracks to mouse, E
DeciGEL	assembler program for stopping 68010 errors, S-E-D
Klock	menu-bar clock and date display, E
life	the game of life, E
TimeSet	intuition-based way to set the time date,
EMEmacs	another Emacs, more oriented to word processing, S-E-D
MyCLI	a CLI shell, works without the Workbench, S-E-D

Texts:

FnctnKeys	explains how to read function keys from Amiga Basic
HackerSIn	explains how to win the game 'hacker'
Ist68010	guide to installing a 68010 in your Amiga
PrinterTip	sending escape sequences to your printer
StartupTip	tips on setting up your startup-sequence file
XfrmrReview	list of Transformer programs that work

Printer Drivers:

Printer drivers for the Canon PJ-1080A, the C Itoh Prowriter, an improved Epson driver that eliminates streaking, the Epson LQ-800, the Gemini Star-10, the NEC 8025A, the Okidata ML-92, the Panasonic KX-P10xx family, and the Smith-Corona D300, with a document describing the installation process.

AMICUS Disk 10 Instrument sound demos

This is an icon-driven demo, circulated to many dealers. It includes the sounds of an acoustic guitar, an alarm, a banjo, a bass guitar, a boink, a calliope, a car horn, claves, water drip, electric guitar, a flute, a harp arpegio, a kickdrum, a marimba, a organ minor chord, people talking, pigs, a pipe organ, a Rhodes piano, a saxophone, a sitar, a snare drum, a steel drum, bells, a vibrophone, a violin, a wailing guitar, a horse whinny, and a whistle.

AMICUS Disk 11

C programs

dirutil	Intuition-based, CLI replacement manager S-E
cpri	shows and adjusts priority of CLI processes, S-E
ps	shows info on CLI processes, S-E
vidtex	displays Compuserve RLE pics, S-E

AmigaBasic programs

pointered	pointer and sprite editor program
optimize	optimization ex ample from AC article
calendar	large, animated calendar, diary and date book program
amortize	loan amortizations
brushtoBOB	converts small IFF brushes to AmigaBasic BOB OBJECTS
grids	draw and play waveforms
hilbert	draws Hilbert curves
madlib	mad lib story generator
mailtalk	talking mailing list program
meadows3D	3D graphics program, from A C™ article
mousetrack	mouse tracking example in hires mode
slot	slot machine game
tictactoe	the game
switch	pachinko-like game
weird	makes strange sounds

Executable programs

cp	unix-like copy command, E
cls	screen clear, S-E
diff	unix-like stream editor uses 'diff' output to fix files
pm	chart recorder performances indicator

Assembler programs

cls	screen clear and CLI arguments example

Modula-2

trails	moving-worm graphics demo
caseconvert	converts Modula-2 keywords to uppercase
Forth	Breshehan circle algorithm example
Analyze	12 templates for the spreadsheet Analyze

There are four programs here that read Commodore 64 picture files. They can translate Koala Pad, Doodle, Print Shop and News Room graphics to IFF format. Getting the files from your C-64 to your Amiga is the hard part.

AMICUS Disk 12

Executable programs

blink	'alink' compatible linker, but faster, E-D
clean	spins the disk for disk cleaners, E-D
epsonset	sends Epson settings to PAR from menu E-D
showbig	view hi-res pics in low-res superbitmap, E-D
speaktime	tell the time, E-D
undelete	undeletes a file, E-D
cnvapidhm	converts Apple][low, medium and high res pictures to IFF, E-D
menued	menu editor produces C code for menus, E-D
quick	quick disk-to-disk nibble copier, E-D
quickEA	copies Electronic Arts disks, removes protection, E-D
txed 1.3	demo of text editor from Micrasmiths,E-D

C programs

spin3	rotating blocks graphics demo, S-E-D
popcli	start a new CLI at the press of a button, like Sidekick, S-E-D
vsprite	VSprite example code from Commodore, S-E-D
AmigaBBS	Amiga Basic bulletin board prog., S-D

Assembler programs

star10	makes star fields like Star Trek intro,S-E-D

Pictures

Mount Mandelbrot	3D view of Mandelbrot set
Star Destroyer	hi-res Star Wars starship
Robot	robot arm grabbing a cylinder

Texts

vendors	Amiga vendors, names, addresses
cardco	fixes to early Cardco memory boards
cinclude	cross-reference to C include files
mindwalker	clues to playing the game well
slideshow	make your own slideshows from the Kaleidoscope disk

AMICUS Disk 13

Amiga Basic programs

Routines from Carolyn Scheppner of CBM Tech Support, to read and display IFF pictures from Amiga Basic. With documentation. Also included is a program to do screen prints in Amiga Basic, and the newest BMAP files, with a corrected ConvertFD program. With example pictures, and the SaveILBM screen capture program.

Routines to load and play FutureSound and IFF sound files from Amiga Basic, by John Foust for Applied Visions. With documentation and C and assembler source for writing your own libraries, and interfacing C to assembler in libraries. With example sound.

Executable programs

gravity	Sci Amer Jan 86 gravitation graphic simulation, S-E-D

Texts

MIDI	make your own MIDI instrument interface, with documentation and a hi-res schematic picture.

AMICUS Disk 14

Several programs from Amazing Computing issues:

Tools

Dan Kary's	C structure index program, S-E-D

Amiga Basic programs:

BMAP Reader	by Tim Jones
IFFBrush2BOB	by Mike Swinger
AutoRequester	example
DOSHelper	Windowed help system for CLI commands, S-E-D
PETrans	translates PET ASCII files to ASCII files, S-E-D
C Squared	Graphics program from Scientific American, Sept 86, S-E-D
crlf	adds or removes carriage returns from files, S-E-D
dpdecode	decrypts Deluxe Paint, remo
ves copy	protection, E-D
queryWB	asks Yes or No from the user returns exit code, S-E
vc	VisiCalc type spreadsheet, no mouse control, E-D
view	views text files with window and
slider	gadget, E-D
Oing, Sproing, Boing! style	yaBoing, Zoing are sprite-based demos, S-E-D

CLIClock, sClock, wClock are window border clocks, S-E-D

Texts

An article on long-persistance phospor monitors, tips on making brushes of odd shapes in Deluxe Paint, and recommendations on icon interfaces from Commodore-Amiga.

AMICUS Disk 15

The C programs include:

'pr'	a file printing utility, which can print files in the background, and with line numbers and control character filtering.
'fm'	displays a chart of the blocks allocated on a disk.
'Ask'	questions an 'execute' file, returns an error code to control the execution in that batch file
'Stat'	an enhanced version of AmigaDOS 'status' command.
'Dissolve'	random-dot dissolve demo displays IFF picture slowly, dot by dot, in a random fashion.
'PopCLI2'	invoke new CLI window at the press of a key.

The executable programs include:

'Form'	file formatting program through the printer driver to select print styles
'DiskCat'	catalogs disks, maintains, sorts,merges lists of disk files
'PSound'	SunRize Industries' sampled sound editor & recorder
'Iconmaker'	makes icons for most programs
'Fractals'	draws great fractal seascapes and mountain scapes.
'3D Breakout'	3D glasses, create breakout in a new dimension
'AmigaMonitor'	displays lists of open files,
memory use,	tasks, devices and ports in use.
'Cosmoroids'	version of 'asteroids' for the Amiga.
'Sizzlers'	high resolution graphics demo written in Modula 2.

Texts:

'ansi.txt'	explains escape sequences the CON: device responds to.
'FKey'	includes template for making paper to sit in the tray at the top of the Amiga keyboard.
'Spawn'	programmer's document from Commodore Amiga, describs ways to use the Amiga's multitasking capabilities in your own programs.

AmigaBasic programs:

'Grids'	draw sound waveforms, and hear them played.
'Light'	a version of the Tron light-cycle video game.
'MigaSol'	a game of solitaire.
'Stats'	program to calculate batting averages
'Money'	"try to grab all the bags of money that you can.

AMICUS 15 also includes two beautiful IFF pictures, of the enemy walkers from the ice planet in Star Wars, and a picture of a cheetah.

AMICUS Disk 16

'juggler'	demo by Eric Graham. a robot juggler bouncing three mirrored balls, with sound effects. Twenty-four frames of HAM animation are flipped quickly to produce this image. You control the speed of the juggling. The author's documentation hints that this program might someday be available as a product.

IFF pictures

parodies of the covers of Amiga World and Amazing Computing magazines.

C programs:

'inputhandler'	example of making an input handler.
'FileZap3'	binary file editing program
'ShowPrint'	displays IFF picture, and prints it.
'Gen'	program indexes and retrieves C structures and variables declared in the Amiga include file system.

Executable Programs:

'FixHunk2'	repairs an executable program file for expanded memory
'ms2smus'	converts Music Studio files to IFF standard 'SMUS' format. I have heard this program might have a few bugs, especially in regards to very long songs, but it works in most cases.
'Missile'	Amiga version of the 'Missile Command' video game,

This disk also contains several files of scenarios for Amiga Flight Simulator II. By putting one of these seven files on a blank disk, and inserting it in the drive after performing a special command in this game, a number of interesting locations are preset into the Flight Simulator program. For example, one scenario places your plane on Alcatraz, while another puts you in Central Park

AMICUS Disk 17

Telcommunications disk whuch contains six terminal programs.

"Comm" V1.33	term prog. with Xmodem, WXmodem,
"ATerm" V7.2	term prog. includes Super Kermit
"VT-100"V2.6	Dave Wecker's VT-100 emulator with Xmodem,Kermit, and scripting
"Amiga Kermit"	V4D(060) port of the Unix C-Kermit
"VTek" V2.3.1	Tektronix graphics terminal emulator based on the VT-100 prog. V2.3 and contains latest 'arc' file compression
"AmigaHost"	V0.9 for Compuserve. Includes RLE graphics abilities & CIS-B file transfer protocol.
"FixHunk"	expansion memory necessity
"FixObj"	removes garbage characters from modem received files
"Txt"	filters text files from other systems to be read by the Amiga E.C.
"addmem"	executeable version for use with mem expansion article in AC v2..1
'arc'	file documentation and a basic tutorial on un 'arc'ing files
"arcre"	for makeing "arc" files E.C.

AMICUS Disk 18

Logo	Amiga version of the populacomputer language, with example programs, E-D
Tv"Text	Demo version of the TV"Text character generator
PageSetter	Freely distributable versions of the updated PagePrint and PageIFF programs for the PageSetter desktop publishing package.
FullWindow	Resizes any CLI window using only CLI commands, E-D
Life3d	3-D version of Conway's LIFE program, E-D
Defdisk	CLI utility to re-assign a new Workbench disk, S-E-D
Calendar.WKS	Lotus-compatible worksheet that makes calendars
SetKey	Demo of keyboard key re-programmer, with IFF picture to make function key labels, E-D
VPG	Video pattern generator for aligning monitors, E-D
HP-10C	Hewlett-Packard-like calculator, E-D
SetPrefs	Change the Preferences settings on the fly, in C, S-E-D
StarProbe	Program studies stellar evolution. C source included for Amiga and MS-DOS, S-E-D
ROT	C version of Colin French's AmigaBasic ROT program from Amazing Computing. ROT edits and displays polygons to create three dimensional objects. Up to 24 frames of animation can be created and displayed. E-D
Scat	Like lng, windows on screen run away from the mouse, E-D
DK	Decays" the CLI window into dust, in Modula 2, S-E-D
DropShadow2	Adds layered shadows to Workbench windows, E-D

AMICUS Disk 19

This disk carries several programs from Amazing Computing. The IFF pictures on this disk include the Amiga Wake part T-shirt logo, a sixteen-color hi-res image of Andy Griffith, and five Amiga Live! pictures from the Amazing Stories episode that featured the Amiga.

Solve	Linear equation solver in assembly language, S-E-D
Gadgets	Bryan Catley's AmigaBasic tutorial,
Household	Bryan Catley's AmigaBasic household inventory program, S-D
Waveform	Jim Shields' Waveform WaveBasic, S-D
DiskLib	John Kennan's AmigaBasic disk librarian program, S-D
Subscripts	Ivan Smith's AmigaBasic subscript example, S-D
String, Boolean	C programs and executables for Harriet Maybeck Tolly's Intuition tutorials, S-E-D
Skinny C	Bob Riemersma's example for making small C programs, S-E-D
COMAL.h	Make C look like COMAL header file,
EmacsKey	Makes Emacs function key definitions by Greg Douglas, S-D
AMon 1.1	Snoop on system resource use, E-D
BTE	Bard's Tale character editor, E-D
Size	CLI program shows the size of a given set of files, E-D
WinSize	CLI window utility resizes current window, S-E-D

AMICUS Disk 20

Compactor, Decoder Steve Michel AmigaBasic tools, S-D

BobEd	BOB and sprite editor written in C,S-E-D
SpriteMasterII	Sprite editor and animator by Brad Kiefer, E-D
BlitLab	Blitter chip exploration C program by Tomas Rokicki, S-E-D
FPic	Image processing program by Bob Bush loads and saves IFF images, changes them with several techniques, E-D
Bankn	Complete home banking program, balance your checkbook! E-D

AMICUS Disk 21

Target	Makes each mouse click sound like a gunshot, S-E-D
Sand	Simple game of sand that follows the mouse pointer, E-D
PropGadget	Harriet Maybeck Tolly's proportional gadget example, S-E
EHB	Checks to see if you have extra-half-bright graphics, S-E-D
Piano	Simple piano sound program
CelScripts	Makes cel animation scripts for Aegis Animator, in AmigaBasic

This disk has electronic catalogs for AMICUS disks 1 to 20 and Fish disks 1 to 80. They are viewed with the DiskCat program, included here.

AMICUS Disk 22

Cycles	Light cycle game, E-D
Show_PrintII	Views and prints IFF pictures, including larger than screen
PrtDrvGen2.3	Latest version of a printer driver generator
Animations	VideoScape animations of planes and boing ball
Garden	Makes fractal gardenscapes
BasicSorts	Examples of binary search and insertion sort in AmigaBasic

AMICUS Disk 23

An AMICUS disk completely dedicated to music on the Amiga. This disk contains two music players, songs, instruments, and players to bring the thrill of playing "Big Sound" on your Amiga

Instruments	a collection of 25 instruments for playing and creating music. The collection ranges from Cannon to Marimba
List INSTR	program to list the instruments DMCS will not load as well as list the origins for any instrument.
Music	a collection of 14 Classical pieces
1812Overture	The 16 minute classical feature complete with Cannon!

Three Amiga Music Players:

SMUSPlay
MusicCraft2SMUS
MusicStudio2SMUS

AMICUS Disk 24

Sectorama	A disk sector editor for any AmigaDOS file-structured device, recover files from a trashed hard disk. By David Joiner of MicroIllusions
Iconize	Reduces the size of IFF images, companion program, Recolor, remaps the palette colors of one picture to use the palette colors of another. Using these programs and a tool to convert IFF brushes to Workbench icons, make icons look like miniatures of the pictures.
CodeDemo	Modula-2 program converts assembler object files to inline CODE statements. Comes with a screen scrolling example
AmiBug	Workbench hack makes the same fly walk across the screen at random intervals. Otherwise, completely harmless.
BNTools	Three examples of assembly language code from Bryce Nesbitt: 1. SetLace.prog to switch interlace on&off. 2. Why, replace AmigaDOS CLI Why 3. Loadlt, prog to load a file into memory until a reboot. (Only the most esoteric hackers will find Loadlt useful.)
Monolace	CLI program resets Preferences to several colors of monochrome & interlace screens. C source is included, works with DisplayPref, a CLI program which displays the current Preferences settings.
BoingMachine	A ray-traced animation of a perpetual motion Boing-making machine, includes the latest version of the Movie program, which has the ability to play sounds along with the animation. By Ken Offer
Daisy	Example of using the translator and narrator devices to make the Amiga talk. It is written in C.
QuickFlix	Script-driven animation and slideshow program flips through IFF images.
BMon	System monitor AmigaBasic program ; perform simple manipulations of memory.
Moose	Random background program, a small window opens with a moose resembleing Bullwinklesaying witty phrases user definable.
DGCS	Deluxe Grocery Construction Set, simple Intuition-based prog for assembling and printing a grocery list.

The Virus Check directory holds several programs relating to the software virus that came to the US from pirates in Europe as detailed in Amazing Computing V2.12. Bill Koester's full explanation of the virus code is included. One program checks for the software virus on a Workbench disk; the second program checks for the virus in memory, which could infect other disks.

AMICUS Disk 25

Nemesis	Graphics demo pans through space towards the mythical dark twin of the sun with wonderful music and space graphics.

The KickPlay directory holds text that describes several patches to the Kickstart disk. For Amiga 1000 hackers who feel comfortable patching a disk in hexadecimal, KickPlay offers the chance to automatically do an ADDMEM for oid expansion memory, as well as the ability to change the picture of the "Insert Workbench" hand. A program is also included for restoring the correct checksum of the Kickstart disk.

KeyBird	BASIC prog edits keymaps, adjust the Workbench keymaps or create your own.

For PDS orders, please use form on page 112

8ColorWB	Modifies the Workbench so three bitplanes are used, icons can have eight colors, instead of four, eight-color icons are included. Public domain program "zapicon" or "brush2icon" converts eight-color IFF brushes to icons, to use Deluxe Paint to make icons for this new Workbench.
BrushIcon	Converts brushes to icons (bizarr docs).
Egraph	Graphing prog reads [x,y] values from a file and displays them on the screen, similar to the same-named Unix program.
Keep 1.1	Message-managing program for telecommunications, lets you save messages from an online transcript to another file, understands the message format of the national networks and several types of bulletin board software. Moves through the transcript and save messages
Kill.fastdir	Speed up directory access, it creates a small file in each directory on a disk which contains the information about the files, will also remove all the "fastdir" files from each directory. by CLImate's authors
The LaceWB program changes between interlace and non-interlace Workbench. Previously, you were forced to reboot after changing Preferences to an interlaced screen. This program flips between the normal and extended screen heights.	
PW_Utility	A shareware utility for ProWrite users, changes margin settings and font types.
Guru	A CLI program, prints out probable causes for Guru mediations; C source included.
DiskWipe	Latest from Software Distillery, removes files from directories or disk drives, much faster than "delete."
Snow	AmigaBasic makes snowflake designs.
Mlist	Mailing list database.
Softballstats	Maintain softball statistics/ team records.
Dodge	Short Modula-2 program moves the Workbench screen around after a period of time, prevents monitor burn-in.

AMICUS Disk 26

Todor Fay's SoundScape module code from his Amazing Computing articles. The source to Echo, Chord, TX, and VU is included. The Lattice and Manx C source code is here, along with the executable modules.	
ImageMaker	Interesting tool edits image structures for C, loads & saves C code directly.
Claz2	Update of prog to convert IFF images to PostScript files for printing on laser printers
SDBackup	Hard disk backup prog with Lempel-Ziv compression to reduce the necessary number of disks.
TCB	Prints information about tasks and processes in the system; assembler source is included.
FunBut	Lets a function key act like a rapid series of left mouse button events.
DC	A handy program for people who use an Amiga 1020 5 1/4 inch drive as an AmigaDOS floppy. A Workbench program that sends a DiskChange signal to the operating system: instead of typing "diskchange df2:" over and over again, just click on the icon. C source included.
System config	File makes screen 80 columns wide of text in the Scribble! word processor.
Dick2Ram	2 programs to move the Scribble! spelling dictionary to and from the RAM disk.
Lexical	Analyzes a text file and gives the Gunning-Fog, Flesch, and Kincaid indices which measure readability.
HexDump	Modula-2 program to display memory locations in hexadecimal.
Tartan	AmigaBasic; design Tartan plaids.
DirMaster	Disk catalog program.
BMP	plays 8SVX sampled sounds in the background while something else is happening in the Amiga, as your Amiga is booting, for example.
ShowPt	CLI program changes your pointer to a given pointer.
AMICUS 26 also has a collection of mouse pointers, & Workbench program to display them	

Fred Fish Public Domain Software

Fred Fish Disk 1:

amigademo	Graphical benchmark for comparing amigas.
amigaterm	simple communications program with Xmodem
balls	simulation of the "kinetic thingy" with balls on strings
colorful	Shows off use of hold-and-modify mode.
dhrystone	Dhrystone benchmark program.
dotty	Source to the "dotty window" demo on the Workbench disk.
treedraw	A small "paint" type program with lines, boxes, etc.
gad	John Draper's Gadget tutorial program
gfxmem	Graphical memory usage display prog.
halfbrite	demonstrates "Extra-Half-Brite" mode, if you have it
hello	simple window demo
ieeeflib	accessing the Motorola Fast Floating Point library from C
palette	Sample prog. to design color palettes.
trackdisk	Demonstrates use of the trackdisk driver.
requesters	John Draper's requester tutorial and example program.
speech	Sample speech demo program.
speechtoy	Stripped down "speechtoy". Another speech demo program.

Fred Fish Disk 2:

alib	Object module librarian.
cc	Unix-like frontend for Lattice C compiler.
dbug	Macro based C debugging package. Machine independent.
make	Subset of Unix make command.
make2	Another make subset command.
microemacs	Small version of emacs editor, with macros, no extensions
portar	Portable file archiver.
xrf	DECUS C cross reference utility.

Fred Fish Disk 3:

gothic	Gothic font banner printer.
roff	A "roff" type text formatter.
tf	A very fast text formatter
cforth	A highly portable forth implementation. Lots of goodies.
xlisp	Xlisp 1.4, not working correctly.

Fred Fish Disk 4:

banner	Prints horizontal banner
bgrep	A Boyer-Moore grep-like utility
bison	CNU Unix replacement 'yacc', not working.
bm	Another Boyer-Moore grep-like utility
grep	DECUS grep
kermit	simple portable Kermit with no connect mode.
MyCLI	Replacement CLI for the Amiga. V. 1.0
mandel	A Mandelbrot set program, by Robert French and RJ Mical

Fred Fish Disk 5

cons	Console device demo program with supporting macro routines.
freemap	Creates a visual diagram of free memory
input.dev	sample input handler, t raps key or mouse events
joystick	Shows how to set up the gameport device as a joystick.
keyboard	demonstrates direct communications with the keyboard.
layers	Shows use of the layers library
mandelbrot	IFF Mandelbrot program
mouse	hooks up mouse to right joystick port
one.window	console window demo
parallel	Demonstrates access to the parallel port.
printer	opening and using the printer, does a screen dump, not working
print.support	Printer support routines, not working.
proctest	sample process creation code, not working
region	demos split drawing regions
samplefont	sample font with info on creating your own
serial	Demos the serial port
singlePlayfield	Creates 320 x 200 playfield
speechtoy	latest version of cute speech demo
speech.demo	simplified version of speechtoy, with IO requests
text.demo	displays available fonts
timer	demos timer.device use
trackdisk	demos trakcdisk driver

Fred Fish Disk 6:

compress	like Unix compress, a file squeezer
dadc	analog clock impersonator
microemacs	upgraded version of microemacs from disk 2
mult	removes multiple occuring lines in files
scales	demos using sound and audio functions
setparallel	Allows changing parallel port parameters
setserial	Allows changing serial port parameters.
sortc	quicksort based sort program, in C
stripc	Strips comments and extra whitespace from C source

Fred Fish Disk 7:

This disk contains the executables of the game Hack V 1.0.1.

Fred Fish Disk 8:

This disk contains the C source to Hack on disk 7.

Fred Fish Disk 9:

moire	Draws moire patterns in black and white
MVP-FORTH	Mountain View Press Forth, version 1.00.03A. A shareware version of FORTH from Fantasia Systems.
proff	a more powerful text formatting program
setface	Prog to toggle interlace mode on and off
skewb	a rubic's cube type demo
sparks	moving snake Graphics demo

Fred Fish Disk 10:

conquest	An interstellar adventure simulation game
dehex	convert a hex file to binary
filezap	Patch program for any type of file.
fixobj	Strip garbage off Xmodem transferred files.
iff	Routines to read and write iff format files.
ld	simple directory program
ls	Minimal UNIX ls, with Unix-style wildcarding, in C
sq,usq	file squeeze and unsqueeze
trek73	Star Trek game
yachtc	Dice game.

Fred Fish Disk 11:

dpslide	slide show program for displaying IFF images with miscellaneous pictures

Fred Fish Disk 12:

amiga3d	Shows a rotating 3 dimensional solid "Amiga sign".
ArgoTerm	a terminal emulator program, written in assembler
arrow3d	Shows a rotating 3 dimensional wire frame arrow.
ld4	directory listing program
IconExec	
SetWindow	two progs for launching progs from Workbench, presently only works under CLI.
SetAlternate	Makes an icon show a second image when clicked once
StarTerm	terminal emulator, with ASCII Xmodem, dialer, more.

Fred Fish Disk 13:

A Bundle of Basic programs, including:

Jpad	toybox	ezspeak	mandlebrot
xmodem	3dsolids	addbook	algebra
ror	amgseq1	amiga-copy	band
bounce	box	brickout	canvas
cardfi	circle	colorcircles	Copy
cubes1	cutpaste	datedogstar	
dragon	draw	dynamictriangle	
Eliza	ezterm	fillibuster	fractal
fscape	gomoku	dart	haiku
hal9000	halley	hauntedM	hidden
join	loz	mandel	menu
minipaint	mouse	Orthello	patch
pena	pinwheel	qboxandom-circles	
Readme	rgb	rgbtest	Rord
sabotage	salestalk	shades	shapes
shuttle			
sketchpad	spaceart	speakspeach	
speecheasy	spell	sphere	
spiral	striper	superpad	suprshr
talk	terminal		
termtest	tom	topography	triangle
wheels	xenos	xmostriper	

(note: some programs are Abasic, most are Amigabasic, and some programs are presented in both languages)

Fred Fish Disk 14:

amiga3d	update of #12, includes C source to a full hidden surface removal and 3D graphics
beep	Source for a function that generates a beep sound
dex	extracts text from within C source files
dimensions	demonstrates N dimensional graphics
filezap	update of disk 10, a file patch utility
gfxmem	update of disk 1, graphic memory usage indicator
gi	converts IFF brush files to Image struct, in Ctext.
pdterm	simple ANSI VT100 terminal emulator, in 80 x 25 screen
shell	simple Unix 'csh' style shell
termcap	mostly Unix compatible 'termcap' implementation.

Fred Fish Disk 15:

Blobs	graphics demo, like Unix 'worms'
Clock	simple digital clock program for the title bar
Dazzle	An eight-fold symmetry dazzler program. Really pretty!
Fish	double buffered sequence cycle animation of a fish
Monopoly	A really nice monopoly game written in AbasiC.
OkidataDump	Okidata ML92 driver and WorkBench screen dump program.
Polydraw	A drawing program written in AbasiC.
Polyfractals	A fractal program written in AbasiC.

Fred Fish Disk 16:

A complete copy of the latest developer IFF disk

Fred Fish Disk 17:

The NewTek Digi-View video digitizer HAM demo disk

Fred Fish Disk 18:

AmigaDisplay	dumb terminal program with bell, selectable fonts
Ash	Prerelease C Shell-like shell program, history, loops, etc.
Browser	wanders a file tree, displays files, all with the mouse
MC68010	docs on upgrading your Amiga to use a MC68010
Multidim	rotate an N dimensional cube with a joystick
PigLatin	SAY command that talks in Pig Latin
Scrimper	Screen image printer
Xlisp1.6	source, docs, and execut for a Lisp interpret.

Fred Fish Disk 19:

BlackJack	text-oriented blackjack game
JayMinerSlides	Slides by Jay Miner, Amiga graphics chip designer, showing flowchart of the Amiga internals, in 640 x 400.
Keymap_Test	test program to test the key mapping routines
LockMon	Find unclosed file locks, for programs that don't clean up.

Fred Fish Disk 20:

AmigaToAtari	converts Amiga object code to Atari form
DiskSalv	program to recover files from a trashed AmigaDOS disk.
Hash	example of the AmigaDOS disk hashing function
Hd	Hex dump utility ala Computer Language magazine, April 86
MandelBrots	Mandelbrot contest winners
MultiTasking	Tutorial and examples for Exec level multitasking
Pack	strips whitespace from C source
PortHandler	sample Port-Handler program that performs. Shows BCPL environment
Random	Random number generator in assembly, f or C or assembler.
SetMouse2	sets the mouse port to right or left
SpeechTerm	terminal Emulator with speech capabilities, XModem
TxEd	Demo editor from Micrasmith's Charlie Heath

Fred Fish Disk 21

This is a copy of Thomas Wilcox's Mandelbrot Set Explorer disk. Very good!

Fred Fish Disk 22

This disk contains two new "strains" of microemacs.

Lemacs	version 3.6 by Daniel Lawrence. For Unix V7, BSD 4.2, Amiga, MS-DOS, VMS. Uses Amiga function keys, status line, execute, startup files, more.
Pemacs	By Andy Poggio. New features include <ALT> keys as Meta keys, mouse support, higher priority, backup files, word wrap, function keys.

Fred Fish Disk 23

Disk of source for MicroEmacs, several versions for most popular operating systems on micros and mainframes. For people who want to port MicroEmacs to their favorite machine.

Fred Fish Disk 24:

Conques	interstaller adventure simulation game
Csh	update to shell on Disk 14, with built in commands,named variables substitution.
Modula-2	A pre-release version of the single pass Modula-2 compiler originally developed for MacIntosh at ETHZ. This code was transmitted to the AMIGA and is executed on the AMIGA with a special loader. Binary only.

Fred Fish Disk 25

Graphic Hack	A graphic version of the game on disks 7 and 8 This is the graphics-oriented Hack game by John Toebes. Only the executable is present.

Fred Fish Disk 26

UniHunk	Processes the Amiga "hunk" loadfiles. Collect code, data, and bss hunks together, allows individual specificatio of code, data, and bss origins, and generates binary file with format reminiscent of Unix "a.out" format. The output file can be easily processed by a separate program to produce Motorola "S-records" suitable for downloading to PROM programmer. By Eric Black.
C-kermit	Port of the Kermit file transfer program and server.
Ps	Display and set process priorities
Archx	Yet another program for bundling up text files and mailing or posting them as a single file unit.

Fred Fish Disk 27

ABdemos	Amiga Basic demos; Caroly Scheppner.
NewConvertFD	creates .bmaps from fd files.
BitPlanes	finds addresses of and writes to bitplanes of the screen's bitmap.
AboutBmaps	A tutorial on creation and use of bmaps.
LoadILBM	loads and displays IFF ILBM pics.
LoadACBM	loads and displays ACBM pics.
ScreenPrint	creates a demo screen and dumps it to a graphic printer.
Disassem	Simple 68000 disassembler. Reads standard Amiga object files and disassembles the code sections. Data sections are dumped in hex. The actual disassember routines are set up to be callable from a user prog so instructions in memory can be disassembled dynamically. By Bill Rogers.
DvorakKeymap	Example of a keymap structure for the Dvorak keyboard layout. Untested but included because assembly examples are few and far between. By Robert Burns
Hypocycloids	Spirograph, from Feb. 84 Byte.
LinesDemo	Example of proportional gadgets to scroll a SuperBitMap.
MemExpansion	Schematics and directions for building your own homebrew 1 Mb memory expansion, by Michael Fellinger.
SafeMalloc	Program to debug 'malloc()' calls
ScienceDemos	Convert Julian to solar and sidereal time, stellar positions and radial velocity epoch calculations and Galilean satellite plotter. By David Eagle.

Fred Fish Disk 28

ABasic games by David Addison: Backgammon , Cribbage, Milestone, and Othello	
Cpp	DECUS 'cpp' C preprocessor, & a modified 'cc' that knows about the 'cpp', for Manx C.
Shar	Unix-compatible shell archiver, for packing files for travel.
SuperBitMap	Example of using a ScrollLayer, syncing SuperBitMaps for printing, and creating dummy RastPorts.

Fred Fish Disk 29

AegisDraw Demo	Demo program without save and no docs.
Animator Demo	Player for the Aegis Animator files
Cc	Unix-like front-end for Manx C
Enough	Tests for existence of system resources, files, and devices
Rubik	Animated Rubik's cube program
StringLib	
Vt100	VT-100 terminal emulator with Kermit and Xmodem protocols

Fred Fish Disk 30

Several shareware programs. The authors request a donation if you find their program useful, so they can write more software.

BBS	an Amiga Basic BBS by Ewan Grantham
FineArt	Amiga art
FontEditor	edit fonts, by Tim Robinson
MenuEditor	Create menus, save them as C source, by David Pehrson
StarTerm3.0	Very nice telecom by J. Nangano

(Fred Fish Disk#30 is free if requested when ordered with at least three other disks from the collection.)

Fred Fish Disk 31

Life	Life game, uses blitter to do 19.8 generations a second.
Mandelbrot	Version 3.0 of Robert French's program.
MxExample	Mutual exclusion gadget example.
RamSpeed	Measure relative RAM speed, chip and fast.
Set	Replacement for the Manx "set" command for environment variables,with improvements.
Tree	Draws a recursive tree, green leafy type, not files.
TxEd	Crippled demo version of Microsmith's text editor, TxEd.
VDraw	Full-featured drawing program by Stephen Vermeulen.
Xicon	Invokes CLI scripts from icon
Ticon	Displays text files from an icon.

For PDS orders, please use form on page 112

Fred Fish Disk 32

Address	Extended address book, AmigaBASIC
Calendar	Calendar/diary program, AmigaBASIC
DosPlus1	First volume of CLI oriented developer tools
DosPlus2	2nd volume of CLI oriented developer tools.
Executables only:	
MacView	Views MacPaint pics in Amiga low or high res, no sample pictures, by Scott Evernden.
Puzzle	Simulation of puzzle with moving squares.
ShowHAM	View HAM pictures from CLI.
Solitaire	ABasiC games of Canfield and Klondike, from David Addison.
Spin3	Graphics demo of spinning cubes, double-buffered example.
Sword	Sword of Fallen Angel text adventure game written in Amiga Basic.
Trails	Leaves a trail behind mouse, in Modula-2

Fred Fish Disk 33

3dstars	3d version of the *stars* program below.
Bigmap	Low-level graphics example scrolls bitmap with ScrollVPort.
Dbuf.gels	Double-buffered animation example for BOBs and VSprites.
DiskMapper	Displays sector allocation of floppy disks.
MemView	View memory in real time, move with joystick.
Oing	Bouncing balls demo
Sproing	Oing, with sound effects.
ScreenDump	Dumps highest screen or window to the printer.
Sdb	Simple database program from a DECUS tape.
Stars	Star field demo, like Star Trek.
TermPlus	Terminal program with capture, library, function keys, Xmodem, CIS-B protocols.
Vt100	Version 2.0 of Dave Wecker's VT-100 emulator, with scripts & function

Fred Fish Disk 34

Alint	Support files for Gimpel's 'lint' syntax checker
Blink	PD 'alink' compaible linker,faster,better.
Browser	Updated to FF 18 'browser', in Manx, with scroll bars, bug fixes.
Btree	b-tree data structure examples
Btree2	Another version of 'btree'
Calendar	Appointment calendar with alarm.
Less	File viewer, searching, position by percent, line number.
NewFonts	Set of 28 new Amiga fonts from Bill Fischer
Pr	Background print utility, style options, wildcards.
Requester	Deluxe Paint-type file requester, with sample.

Fred Fish Disk 35

ASendPacket	C example of making asynchronous I/O calls to a DOS handler, written by C-A
ConsoleWindow	C example of getting the Intuition pointer a CON: or RAW: window, for 1.2, by C-A.
DirUtil	Walk the directory tree, do CLI operations from menus
DirUtil2	Another variant of Dirutil.
FileRequester	Lattice C file requester module, with demo driver, from Charlie Heath.
MacView	Views MacPaint pictures in Amiga low or high res, with sample pictures, by Scott Evernden.
Plop	Simple IFF reader program
PopCLI	Sidekick-style program invokes a new CLI, with automatic screen blanking.
QuickCopy	Devenport disk copiers duplicate copy-protected disks.
ScrollPf	Dual playfield example, from C-A, shows 400 x 300 x 2 bit plane playfield on a 320 x 200 x 2 plane deep playfield.
SendPacket	General purpose subroutine to send AmigaDos packets.
SpriteMaker	Sprite editor, can save work as C data structure. Shareware by Ray Larson.
Tracker	Converts any disk into files, for electronic transmission. Preserves entire file structure. Shareware by Brad Wilson.
TriClops 3-D	space invasion game, formerly commercial, now public domain. From Geodesic Publications.
Tsize	Print total size of all files in subdirectories.
Unifdef	C preprocessor to remove given #ifdef'd sections of a file, leaving the rest alone. By Dave Yost
Vttest	VT-100 emulation test program. Requires a Unix system.

Fred Fish Disk 36

Acp	Unix-like 'cp' copy program
Clock	Updated version of clock on disk 15.
Csh	Manx 'csh'-like CLI, history, variables, etc.
DietAid	Diet planning aid organizes recipes, calories
Echo	Improved 'echo' command with color, cursor addressing
FixHunk	Fixs programs to let them run in external memory.
Fm	Maps the sectors a file uses on the disk.
KickBench	Docs, program to make a single disk that works like a Kickstart and Workbench.
Lex	Computes Fog, Flesch, and Kincaid readability of text files.
TunnelVision	David Addison ABasic 3D maze perspective game.
Vc	Visicalc-like spreadsheet calculator program.
Vt100	Version 2.2 of Dave Wecker's telecom program
YaBoing	Oing! style game program shows sprite collision detects

Fred Fish Disk 37

This disk is a port of Timothy Budd's Little Smalltalk system, done by Bill Kinnersley at Washington State University.

Fred Fish Disk 38

CSquared	Sep 86 Sci American, Circle Squared algorithm
FixObj	Strips garbage off Xmodem transfered object files
Handler	AmigaDOS handler (device) example from C-A
Hp-10c	Mimics a HP-10C calculator, written in Modula-2
IFFEncode	Saves the screen as an IFF file
IffDump	Dumps info about an IFF file
Jsh	BDS C-like CLI shell
NewStat	STATUS-like program, shows priority, processes
Reversi	Game of Reversi, version 6.1
UUdecode	Translate binary files to text, Unix-like programs
Vdraw	Drawing program, version 1.14
VoiceFiler	DX MIDI synthesizer voice filer program
Window	Example of creating a D OS window on a custom screen

Fred Fish Disk 39

AnsiEcho	'echo', 'touch', 'list', 'cls' written in assembler.
Display	Displays HAM images from a ray-tracing program, with example pictures.
Driver	Example device driver source, acts like RAM: disk
Xlisp	XLisp 1.7, executable only

Fred Fish Disk 40

Ahost	Terminal emulator with Xmodem, Kermit and CIS B protocols, function keys, scripts, RLE graphics and conference mode.
AmigaMonitor	Dynamically displays the machine state, such as open files, active tasks, resources, device states, interrupts, libraries, ports, etc.
Arc	Popular file compression system, the standard for transiting files
AreaCode	Program that decodes area codes into state and locality.
Blink	'alink' replacement linker, version 6.5
Cosmo	An 'asteriods' clone.
Dg210	Data General D-210 Terminal emulator
DirUtil	Windowed DOS interface program, V 1.4
DOSHelper	Windowed AmigaDOS CLI help program
PagePrint	Prints text files with headers, page breaks, line numbers
PopCLI	Starts a new CLI with a single keystroke, from any program, With a screen-saver feature. Version 2, w8burce.
SpriteEd	Sprite Editor edits two sprites at a time
X-Spell	Spelling checker allows edits to files

Fred Fish Disk 41

AmigaVenture	Create your own text adventure programs in AmigaBasic.
Csh	Version 2.03 of Dillon's C sh-like shell. Executable only
Dbug	Macro based C debugging packageupdate to FF #2
DualPlayField	example from CBM, update to Intuition manual
GetFile	Heath's file requester, with source
LatXref	Cross reference of Lattice 3.10 header files
Lines	Line drawing demo program
SetFont	Changes font used in a CLI window
Vt100	Version 2.3 of the VT-100 terminal program.

Fred Fish Disk 42

This disk contains an Amiga version of MicroGNUEmacs.

Fred Fish Disk 43

BasicBoing	AmigaBasic program demos page flipping of a 3D cube
Bbm	Demo copy of B.E.S.T. Business Management System.
BbsList	A list of Amiga Bulletin Board Systems
Cc	C compiler frontends for Manx and Lattice C
Copper	A hardware copper list disassembler
InstIFF	Converts Instruments demo sounds to IFF sampled sounds
PopColours	Adjust RGB colors of any screen
SpriteClock	Simple clock is displayed on a sprite above all screens
ST Emulator	Non-serious Atari ST emulator
WBrun	Lets Workbench programs be run from the CLI
Wild	Two Unix shell style wild card matching routines

Fred Fish Disk 44

Icons	Miscellaneous icons
NewIFF	New IFF material from CBM for sampled voice and music files
RayTracePics	The famous ray-tracing pictures, from FF#39, now converted to IFF HAM format for *much* faster viewing.
ViewILBM	Displays normal and HAM ILBM files

Fred Fish Disk 45

Clue	Clue board game
Make	Another 'make', with more features
Pictures	Miscellaneous pictures
Update	Updates older disk with newer files fromanother disk
Wherels	Searches a disk for files of given name

Fred Fish Disk 46

Asm	Shareware 68010 macro assembler, ROM Kernal Manual compatible
CheckModem	'execute' file program detects presence of modem
Egad	Gadget editor from the Programmers Network
Jive	Transforms a file from English to Jive.
My.lib	A binary only copy of Ma tt's alternate runtime library. Author: Matt Dillon
ProffMacros	Subset Berkeley 'ms' and 'mm' macros for 'proff'
ValSpeak	Transforms a file from English to Valley Speak.

Fred Fish Disk 47

3D-Arm	Simulation of a robotic a rm, very good graphics, teaching tool, including C source.
Juggler	Eric Graham's stunning HAM animation of a robot juggler
VT-100	Version 2.4 of Dave Wecker's terminal emulator, with Xmodem and Kermit file transfer protocols

Fred Fish Disk 48

Bru	Alpha version of a hard disk file archiver
Comm	Version 1.30 of a terminal emulator with phone directories
Csh	Version 2.04 of Matt Dillion's Unix 'csh'-like CLI replacement, including Lattice & Manx C source
Diskperf	Disk benchmark program for Unix and Amiga
Du	Computes disk storage of a file or directory
MemWatch	Program to watch for programs that trash low memory. It attempts to repair the damage, and puts up a requester to inform you of the damage. From the Software Distillery.
Profiler	A realtime execution profiler for Manx C programs. Includes C source.

Fred Fish Disk 49

Cycloids	Update of electronic spirograph from disk 27
DirUtil	Enhanced version of DirUtil from disk 35
MultiDef	Scans a set of object modules and libraries searching for multiply defined symbols
MyUpdate	Disk update utility with options for stripping comments from C header files, and interactive verification of the updating process
Plot	Computes and displays 3 dimensional functions in hires
Polygon	Moire type pattern generator with color cycling
QMouse	Queries whether a mouse button is pressed. This can give a return code that can customize a startup-sequence based on whether a mouse button was pressed.
Touch	Example of setting the datestamp on a file, using a technique from Commodore-Amiga
Trees	More extensive version of the trees program on Disk 31

Fred Fish Disk 50

Asm	Version 1.1 of a shareware 68000 macro assembler, compatible with the Metacomco assembler. This includes an example startup module and more Motorola mneumonics.
BreakOut	A brick breakout game, uses 3-D glasses
DiskZap	Version 1.1 of a program to edit disks and binary files
FirstSilicon	A smart CLI replacement with full editing and recall of previous commands
Missile	A Missile Command-type game, with sound, in assembler
PerfectSound	Sound editor for a low-cost sound digitizer
Sizzlers	Graphics demos
UnixArc	Ver of 'arc' for Unix System V machines,in C
Wombat	Version 3.01 of Dave Warker's terminal emulator

Fred Fish Disk 51

Bison	GNU for Unix 'yacc', working update to FF4
Compress	Update to the file compression program on Disk 6
Cos	"Wheel of Fortune"-type game in AmigaBasic
DifSsed	Unix-like 'dif' and 'ssed' for finding the differences between two files, and then recreating the other, given one file, and the list of differences.
Sq, Usq	Portable versions of the CP/M squeeze and unsqueeze

Fred Fish Disk 52

Assign	Replacement for AmigaDOS 'assign' command in C
Fractal	Makes random fractal terrains
Poly, HAMPoly	Workbench-type demos for making polygons in lores and HAM
MxGads	Example of mutual exclusion gadgets with GadgetText
Tek4010	Tektronix 4010 terminal emulator
VDraw	Versions 1.16 and 1.19 of a Deluxe Paint-like drawing program

Fred Fish Disk 53

Animations	Demo animations with player program for Aegis Animator
ARCre	Creates rename scripts for files with long names, so they can be easily 'arc'ed and un'arc'ed.
ARP	Preliminary AmigaDOS replacements for 'break', 'cd', 'chmod', 'echo', 'filenote' and ' makedir'
Compiler	Not fully ported to the Amiga, this is a 68000 C compiler. It wil produce simple assembly language output, but needs a lot of work.
Spreadsheet	Update with source of the 'vc' spreadsheet on disk 36
TarSplit	Port of program to split Unix 'tar'archives
UUencode	Utilities to encode and decode binary files for ASCII transmission, expanding them by 35%

Fred Fish Disk 54

Hanoi	Solves Towers of Hanoi Problem in it's own Workbench window. by Ali Ozer
ISpell	Port of a Unix screen oriented, interactive spelling checker. (Expansion RAM required) by Pace Willisson
Ing	A Screen of lots of bouncing little windows by Leo 'Bols Ewhac' Schwab
Lav	Displays number of tasks in run queue, averaged over last 1, 5, and 15 minute periods. by Willaim Rucklidge
MIDITools	Programs to play/record through the MIDI I/F. by Fred Cassirer
MoreRows	Program to make the Work Bench Screen larger than normal. by Neil Katin and Jim Mackraz
Tilt	Program to make your Amiga look like it didn't pass vibration testing. by Leo ' Bols Ewhac' Schwab

Fred Fish Disk 55

Csh	V2.05 of Matt Dillon's csh like shell (Modified for Manx C). by Matt Dillon, Modified by Steve Drew
NewStartups	New C Startup modules:
AStartup.asm	with 1.2 fixes and better quote handling.
TWStartup.asm	opens a stdio window, using user specs. by Commodore, posted to BIX by Carolyn Schepper
Palette	Change another program's screen colors. by Carolyn Schepper
PipeDevice	Allows the standard output of one process to be fed to the standard input of another. by Matt Dillon
ScreenSave	Save a normal or HAM mode screen as an IFF file. by Carolyn Schepper
ShanghaiDemo	Demo of the Activision game Shanghai.
SoundExample	A double buffered sound example for Manx C. by Jim Goodnow
Vsprites	A working vsprite example. by Eric Cotton
Vt100	V2.6 of Dave's Vt100 terminal emulator with kermit and xmodem. by Dave Wecker

Fred Fish Disk 56

ClipBoard	Clipboard device interface routines, to provide a standard interface. by Andy Finkle
ConPackets	Demos the use of DOS Packets, ConUnit, etc. by Carolyn Schepper
GetDisks	Program to find all available disk device names and return them as an exec list. by Philip Lindsay
GetVolume	Program to get volume name of the volume that a given file resides on. by Chuck McManis
Icon2C	Reads an icon file and writes out a fragment of C code with the icon data structures. by Carolyn Schepper
MergeMem	Program to merge the MemList entries of sequentially configured RAM boards. by Carolyn Schepper
mCAD	An object oriented drawing program, V1.1 by Tim Mooney

Fred Fish Disk 57

Replaced by FF97 Due to Copyright problems

Fred Fish Disk 58

ASDG-rrd	Extremely useful shareware recoverable ram disk. by Perry Kivolowitz
BigView	Displays any IFF picture, independent of the physical display size, using hardware scroll. by John Hodgson
EGraph	Reads pairs of x and y value from a list of files and draws a formatted graph. by Laurenece Turner
HyperBase	Shareware data management system. V1.5
MemClear	Walks through the free memory lists, zeroing free memory along the way. by John Hodgson
NewZAP	A third-generation multi-purpose file sector editing utility. V3.0 by John Hodgson
RainBow	A Maurauder-Style rainbow generator. by John Hodgson
SMUSPlayers	Two SMUS plays, to play SMUS IFF music formatted files. by John Hodgson
View	A tiny ILBM viewer by John Hodgson
WBdump	JX-80 optimized workbench printer that does not use DumpRPort. by J. Hodgson

Fed Fish Disk 59

Browser	Update to browser program on disks 18 and 34. S-E
Browser2	Another different browser program. E
Clock	Clock program with fonts, colors. E
Dme	Dillon text editor V1.22 for programmers,ED
DropCloth	Puts pattern on Workbench backdrop,ED
DropShadow	Puts shadows on Workbench windows,E-D
FixWB	Similar to DropCloth, but doesn't work yet. S-D
mCAD	Object-oriented drawing program, version 1.2.2. Much improved over disk 56.
Robotroff	Demo of animated pointers on Workbench. S-E-D
Supermort	General compounding/amortization loan calculator. E-D

Fred Fish Disk 60

Various shareware and freeware programs

Blitz	Memory resident file viewer. Very fast. E-D
BlitzFonts	Makes text output faster. E-D
HandShake	Terminal emulator with VT52/VT100/ VT102support. E-D
Med	Mouse-driven text editor version 2.1. E-D
PrtDrvGen	Generates printer drivers, version 1.1.S available from author. E-D
Show	Slideshow-like IFF viewer, V2.1. E-D
Uedit	Customizable text editor V2.0. E-D
Ueturbo	Example Uedit setup macros. S-E-D

Fred Fish Disk 61

ATPatch	Patches Transformer to work under AmigaDOS 1.2. S-E-D
FillDisk	Writes zeroes to free blocks on a disk for security. S-E-D
LPatch	Patch for programs that abort when loading under AmigaDOS 1.2. S-E-D
MicroEmacs	Conroy MicroEmacs V3.8b, newer than disk 22. S-E-D
PearlFont	Like Topaz, but rounded edges.
Terrain	Generates fractal scenery. S-E-D
VSprites	Makes 28 Vsprites, from P&E1Book.

Fred Fish Disk 62

This is a port of the Unix game 'Hack', by the Software Distillery, version 1.0.3D.

Fred Fish Disk 63

This is a port of the Unix game 'Larn', by the Software Distillery, version 12.0B.

Fred Fish Disk 64

This is an offical IFF specification disk from Commodore, an update to disk 16.

Fred Fish Disk 65

Bawk	Unix text processor, like 'awk'. Doesn't work, but source is included. S-E-D.
MWB	Example of rerouting Workbench window open calls to another custom screen. Version 1.01, S-E-D
CloseWB	Example for closing a custom Workbench screen. S-E-D
Cookie	Generates one-line fortune-cookie aphorisms. S-E-D
JTime	Build-your-own mouse port clock.
MenuBuilder	Creates C source files for menus, based on text descriptions. S-E-D.
NewPackets	CBM tutorial on new packets and structures in AmigaDos 1.2.
PascalToC	Pascal to C translator, not so great. S-E-D
Prep	'ratfor'-like FORTRAN preprocessor. S-E-D
RunBack	Starts programs from CLI, allowing CLI window to close. E-D

For PDS orders, please use form on page 112

SunMouse This program automatically clicks in windows when the mouse is moved over them. V1.0, E-D

Fred Fish Disk 66

AmScsi Preliminary plans for a SCSI disk controller board.
Asm68k Macro assembler, version 1.0.1. E-D
Assigned Example for avoiding DOS insert-disk requester, by scanning the list of 'assign'ed names. S-E-D
Dk Pretends to eat away at CLI window. S-E-D
Flip Flips whole screen as a joke. S-E-D
Foogol Foogol cross-compiler generates VAX assembly code. S-E-D
Free Prints amount of free space on all drives.S-E-D
MallocTest malloc/free memory test program. S-E-D
Melt Pretends to melt the screen. S-E-D
Nart Graphic flying string demo. S-E-D
Purty Easy way to set printer attributes from Workbench. E-D
RayTracer Simple ray tracing program. E-D
SendPackets Updated CBM examples of packet routines on disk 35. S-E-D
SnapShot Memory resident screen dump. E-D
TagBBS Shareware BBS system, version 1.02.

Fred Fish Disk 67

AmCat Shareware disk cataloging program.
AmigaSpell Shareware Intuition spelling checker, V2.0. E-D
Bouncer 3-D bouncing ball written in MultiForth, SED
Comm Terminal program version 1.33, E
Dux5 Another version of DirUtil. S-E-D
HexCalc Hex, octal, & decimal calculator. E-D
Icons Various big and alternate image icons.
Mandala Mandala graphics and sound. E
PersMail Demo shareware personal file manager.
RSLClock Menu bar clock version 1.3. E-D
RTCubes Graphics demo of 3D cubes. E-D
Wheel "Wheel of Fortune"-type game,AmigaBASIC

Fred Fish Disk 68

This is version MG 1b of the MicroGNUEmacs. Source and executable areincluded, as well as source for other computers besides the Amiga.

Fred Fish Disk 69

Asm68k Macro assembler, v1.0.3, E-D
BlitLab Blitter exploring program, in C, S-E-D
Conman Replacement console device handler adds editing and history to any application that uses CON:, v0.9, E-D
Console Replacement console routines, in C, S-E-D
Dk Decays the screen bit by bit, update to disk 66, in Modula-2, S-E-D
Frags Displays memory fragmentation by listing the size of free memory blocks, in C, S-E-D
IconType Change the type of an icon, in C, S-E-D
Make 'make' in Manx C, S-E-D
MonProc Monitors processes for packet activity, in C,S-ED
MouseClock Mouse pointer into a digital clock,in C,SED
Sb Browses system structures, from Transactor magazine, v1.0, in C, S-E-D
Spew Generates 'National Enquirer'-type headlines from rules file. In C,S-E-D
Spool Three programs to demonstrate multitasking & spooling in a printer spooler. In C, v1.2, S-E-D
Wc Counts words ala Unix 'wc', but faster, in C,S-ED

Fred Fish Disk 70

This is a disk of shareware programs.

AmigaMonitor Explores state of the system, v1.13
Arc Standard file compressor andl librarian, v0.23, a port of MS-DOS v5.0. E-D
BlackBook Phone book program.
DoTil Intuition-driven file manipulator program,v2.0.
GravityWars Game of planets, ships and black holes,v1.03.
Jobs Alternate user interface to CLI and WB, v2.1.
Lens Magnifies area around mouse, shows it in a window, v1.0.
Life-3d 3D version of the classic cellular-automaton game, v1.2.
Logo Logo language interpreter
SetKey Demo keymap editor, v1.0
Vpg Makes displays for aligning video monitors, v1.0.

Fred Fish Disk 71

AirFoil Makes airfoils using the Joukowski transformation, in C, S-E-D
Amiga Basic Miscellaneous programs including 3D plot program, a kaleidoscope, C-A logo drawing program file comparison utility string search program, S-E-D
Blocks A variation of "lines", but with variable color blocks. E-D
Comm Great terminal program, v1.34, E-D
DiskX Utility for exploring file system.E-D
Fpic Simple image processing program that operates on IFF pictures, with several filters, merging images, E-D
IconMk Makes icons for files , v1.2a, E-D
Icons New icons
NewFonts Two new fonts; 'shalt18', an electronic ciruit element font, and 'ibm5', a PC-like font.
PetCLI An AmigaBASIC CLI shell program.
PWDemo Demo of the commercial product PowerWindows,v1.2. It aids creation of custom windows, menus, and gadgets, giving C or assembly source. E-D
Rot Creates and animates 3-D objects, v0.5, E,D
TimeSet Sets time from Workbench, E-D

Fred Fish Disk 72

This is a disk of IFF pictures.

Fred Fish Disk 73

Add Customizes existing program menus with Amiga-key shortcuts. Also includes 'until', which waits until a given window is created. Shareware, in C, S-E-D.

AutoIconOpen Fools WB into thinking mouse has double-clicked icons. In C, S-E-D
Dio Generic Exec device interface code for opening libraries, getting multiple I/O channels, asynchronous operations, etc. In C, S-E-D.
Dissolve Slowly displays IFF files, ala Nov 86 Dr. Dobb's program. In C, S-E-D
DTerm Flexible, reprogrammable terminal program v1.10, E-D
Expose Re-arranges windows so that at least one pixel of menu bar gadgets are exposed. InC, S-E-D.
Lit Scans a text file, converts to C-style printable strings.C,v2.0, S-E-D
Lmv "Long Movie", program views series of IFF picts in quick succession, upto 19 fps. Shareware, E-D
MouseOff Mouse pointer disappears after ten seconds of non-use. In C, S-E-D
ParOut Examples of controlling parallel port with resources instead of the PAR: device.In C,S-E-D
PenPalFont Child-like font.
RunBackGround Similar to RunBack on disk 66, runs program from the CLI allowing the CLI window to close. In C,S-E-D
SnapShot Screendump utility,update FF 66.E-D
TypeAndTell Example installs a device handler before Intuition, and speaks each key as it is pressed. In C and assembler, S-E-D
Xplor Prints info about system lists, in assembler,S-E-D

Fred Fish Disk 74

Cled Edits and recalls CLI commands, v1.3, E-D
Control Intercepts graphic printer dump calls and accesses color map, width,and screen resolution. C,S-E-D
Dme Simple WYSIWYG text editor for programmers,v1.25. Update of FF 59.E-D
DropShadow WB dropshadows, v2.0.UpdateFF59. E-D
Funds AmigaBASIC prog tracks mutual or stocks-D
Less Text viewing program, like Unix "more", v1.1, update to disk 34. S-E-D
Makemake Scans C source files and constructs a vanilla 'makefile' in the current directory. S-E-D
mCAD Object-oriented drawing prog. v1.2.4, update to FF 59.Shareware, E-D
Random Simple random number generator in C. S-E-D
TDebug Monitors devices by intercepting Exec SendIO() and DoIO() vectors, in C, v1.0, S-E-D
Units Converts measurements in different units, includes "chart" option, in C, S-E-D
XCopy Replacement for AmigaDOS 'copy', doesn't change the date, uses Unix wildcards. E-D

Fred Fish Disk 75

Bezier Play with Bezier curves points and granularity, S-E-D
BSplines Play with b-splines, as above, S-E-D
Comm C source for Comm terminal program v1.34. S-E-D
Copy Replacement 'copy' command v1.0, preserves date, in C, S-E-D
Diff Simple 'diff' in C, S-E-D
DuM2 Another DirUtil in Modula-2, v1.5, S-E-D
Eless Fast 'dir' program in C, S-E-D
Fd Faster 'eless' in C, S-E-D
HardCopy Sends a transcript of a CLI session to a file, in C, S-E-D
MouseOff Update FF73, turns off mouse pointer, S-E-D
SetFont Changes the font in a Workbench screen, v2.0, S-E-D
SpeedDir Another fast 'dir', in assembler, S-E-D

Fred Fish Disk 76 & 77

These are disks 1 and 2 of Chris Gray's Draco distribution for the Amiga. Draco is a compiled, structured language reminiscent of both C and Pascal. A full interface to AmigaDOS and Intuition is supplied. Be sure to get both disk 76 and 77.

Fred Fish Disk 78

Cycles Cycle game like "Tron", v1.0, E-D
EOMS Experts Only Mercenary Simulator game, E-D
MandelVroom Mandelbrot generator with enhanced palette controls, fixed/floating point, presets, v1.50, in Manx C, S-E-D

Fred Fish Disk 79

AsmTools CLI tools in assembler: echo, loadit, mounted, setlace, why; S-E-D
AssignDev Give devices multiple names, in C, S-E-D
AuxHandler Example of a dos handler that allows use of a CLI via the serial port. Includes source. Author: Steve Drew
Cmd Redirects printer output to a file, in C, S-E-D
Info AmigaDOS 'info' replacement, in C and assembler, S-E-D
Kill Removes a task and its resources, in C,S-E-D
M2Error Displays errors from TDI Modula-2 compiles, S-E-D
MonProc Update to process packet prog. from FF69,in C,S,E,D
Mounted Program for testing if a drive is present, in a script In C, S-E-D
Nro Another 'roff'-style text formatter, in C, S-E-D
ParTask Finds parent task, in C, S-E-D
QueryAny For scripts, asks a question, accepts Y/N, gives return code. In assembler, S-E-D
ScnSizer Resets pref settings for screen size, in C,SED
SharedLib Example, shared lib, in C & assembler, S-E-D
Task Simple CreateTask() example in C, S-E-D
Uw Unix Windows client v1.0, in C, S-E-D
Who Lists tasks on ready and wait queues, in C, S-E-D

Fred Fish Disk 80 (see Fred Fish 90)

Fred Fish 80 has been withdrawn due to copyright problems.

Fred Fish Disk 81

Asm68k V1.1.0 of a macro assembler
AutoFacc Shrinks the FACC window and moves it to the back
Brushes 53 custom IFF brushes of electronic symbols
CheckIFF Checks structure of an IFF file CledV1.4 update FF74 of a simple CLI
Conman Replaces console handler to add editing and history to many programs
Fonts Miscellaneous fonts
Icon V6.0 of the icon programming language
KeyLock Freezes the keyboard and mouse until pass word entered.

ScatDisplay hack created from "ing"
Smush Smushes an IFF file.
Target Each mouse clikc becomes a gunshot

Fred Fish Disk 82

Adventure Port of the classic Crowther and Woods game
AmicTerm V0.50 of a telecommunications program, with scripts, redial, beeps, enhanced file requester
D2D-Demo Demo version of Disk-2-Disk by Central Coast Software
DX-Synth Voice filer program for Yamaha DX series synthesizers, update to disk 38
DiskMan V1.0 of another DirUtil program
Icons Miscellaneous new icons
Panl Universal MIDI patch panel, v1.2
Rocket Another Workbench hack, plays Lunar Lander
Sand Game of sands following your pointer.

Fred Fish Disk 83

This disk contains a demo version of TeX from N Squared. It is limited to small files, and the previewer can only display ten pages or less, and only a small number of fonts are provided.

Fred Fish Disk 84

AudioToolsPrograms from Rob Peck's July/August Amiga World article
BlitLab Blitter experimentation program, V1.2, update to FF69
Ed Simple editor, similar to Unix 'ed', based on the editor in Software Tools.
GravityWars Game of planets, ships and black holes, v1.04, update to disk 70.
HunkPad Adds legal padding to executables for Xmodem transmission.
PipeHandler An AmigaDOS pipe device which supports named pipes and taps. V1.2
PopCLI V3.0 of a hot-key to invoke a CLI window, with screen blanker, update to disk 40.
Requester Update FF34, file requester similar to DPaint.
ScottDevice V33.1 of a 'mount'able MicroForge SCSI driver.
Viacom Another Schwab hack, makes TV-like static on screen.Parody

Fred Fish Disk 85

Csh V2.06 of Dillon's 'csh'-like shell
FileReq Source to wildcard file requester
Hide Hides expansion memory from programs
ImageTools Shareware tools to manipulation IFF images
LowMem ServerShared library to aid in low memory situations
Plot6 A star plotting program with source.
RawIO Example of setting raw mode on standard input
Rocket Lunary Lander for Workbench, with source.
VMore "more"-like text viewing utility, v1.0 SE
Vnews Simple Unix news reader.

Fred Fish Disk 86

AutoPointAuto-selects window under the mouse pointer, with screensaver.
ClickToFront Double-clicks in window brings it to front, v1.1, S-E-D
Cmd V3.0 of a tool to redirect printer output toa file.
FileIISG-Demo Demo of Softwood File IIsg, a database manager with sound and graphics.

Fred Fish Disk 87

AdvSys Adventure system from Byte May 1987, v1.2 E-D
AutoIconOpen Fools Workbench to open disk icons, V1.2 update to disk 73, S-E-D
Claz Converts IFF files to PostScript, V2.0, SED
Commodi tiesMackraz's Commodities Exchange, an exec library to manage input handler, v0.4
Diff Update to disk 75 of Unix-like 'diff', S-E-D
Dme V1.27 of Dillon's text editor, update FF74,E-D
DropShadow V2.0 of prog. that puts shadows on Workbench, S-E-D
Elib Shared library example in Manx C.
ID-Handler An AmigaDOS device handler generates unique identifiers, V1.0, S-E-D
Install Alternate AmigaDOS 'install' programs,SED
MemWatch Waits for low memory trashing, V2.0, SED
MovePointer Moves pointer to given location, S-E-D
MoveWindow Move window to given location, S-E-D
MunchingSq Munching Squares hack, S-E-D
PalTest Test to see if this is a PAL machine, S-E-D
Sc Generates random scenery, S-E-D
Tek4695 Tek4695 printer driver
WBDualPF Example of dual-playfield screen, update FF41, S-E-D
WarpText Fast text rendering routines, S-E-D
Yailf rExample IFF reader, S-E-D
Zoo A file archiver like *arc*, v1.42A, E-D

Fred Fish Disk 88 (see Fred Fish 89)

FF Disk 88 has been removed due to copyright problems

Fred Fish Disk 89 (replaces Fred Fish 80)

DirMaster Disk catalogue program, V1.0a, E-D
FuncKey Shareware function key editor, V1.01, E-D
MFF-Demo Demo of MicroFiche Filer database prog
ScreenShift Adjust screen position ie Preferences,SED
Snake Bouncing squiggly lines demo, S-E-D
AutoEnguirer screen contraption requester improvement S-E-D
DemoLition Display Hack S-E-D

Fred Fish Disk 90 (replaces Fred Fish 80)

AmiGazer Night sky viewer of 1573 stars, set date, time, day. E-D
CardFile AmigaBasic card file study aid. E-D
Conman Console handler replacement gives line editing and history to most progs, v0.98,ED
IMandelVroom Slight update to disk 78 Mandelbrot program, E-D
NewDemos Replacements for lines and boxes demos that take less CPU time, E-D
Othello Game of Othello, E-D
PrinText Displays text files with gadgets, speech, IFF display, v1.2, E-D
PrtDrvGen Automatic printer driv. generator,v2.2b,ED
RainBench Cycles colors of WB backdrop or text. ED
ShortCut Makes single-key shortcuts for entering commonly typed CLI commands,&custom macros. E-D
ShowPrint Displays and prints all sizes of IFF pictures & controls printer output styles, v2.0 E-D
Sizzlers Graphics demos, v1.7.0, E-D
Timer Small Workbench timer counts time and $/ minute, E-D

Fred Fish Disk 91

Adventure Definition Language (ADL) a superset of an older language called DDL by Michael Urban, Chris Kostanick, Michael Stein, Bruce Adler, and Warren Usui. ADL enhancements by Ross Cunniff. Included are sources to the ADL compiler, interpreter, and debugger. Binaries combined by Ross with Lattice 3.03. CLI enviornment only. Documentation is available from the authors.

Fred Fish Disk 92

As6502 portable 6502 assembler, C source, by J. Van Ornum, Amiga port by Joel Swank
Bawk Text processor update from FF65 Inspired by UNIX awk. Searches files for patterns, performs actions based on patterns. By Bob Brodt; Amiga port by Johan Widen
HunkPad update of FF84 version. by J. Hamilton,pads an object file to a multiple of 128 bytes for better xmodem transfer. S E
Less Like Unix "more", better, version 1.2 update of FF74. Scrolls Back and f orward. S E by Mark Nudelman, Amiga port by Bob Leivian.
Ndr Library that implements the 4BSD unix dir access routines by MikeMeyer. S
Parse Recursive descent expression parcer, computes, and prints expressions. includes transcendental function support. c Source included. by J Olsen
Shar Two programs to pack and unpack shell archives includes C source. by Fabbian G. Dufoe
SmallLib 8 times smaller Amiga.lib replacement, binary only. by Bryce Nesbitt
UUencode Encode/decode binary files for e mail or text-only methods. Update of FF53, includes checksum technique, compatible with older versions, plus transparent to older versions options. By Mark Horton, modified by Alan Rosenthal & Bryce Nesbitt.

Fred Fish Disk 93

Dme Version 1.27 WYSIWYG programmer editor. Not a word processor. Includes key mapping, fast scro-lling, title-line statistics, multiple windows, ability to iconify windows. Update of FF87, SE. by Matt Dillon
MicroEmacs Version 3.8, update to FF61 includes source. Orig by Dave Conroy modifications by Daniel Lawrence

Fred Fish Disk 94

AudioTools Demo programs from Rob Peck's July/Augustt issue of AmigaWorld on accessing the audio device. V2,update of FF84. S. by Rob Peck
ClickUpFront Similar in function to ClickToFront prog (FF86), bring windows tofront by clicking on any part of them. V 1.0. by Davide Cervone SE
HeliosMouse Automatically activate a window simply by moving the mouse pointer into the window. V 1.0. Includes source. By Davide Cervone
IFF2Ps Convert any IFF file to postscript for printing or viewing on a postscript compatible device. Version 1.2, by William Mason and Sam Paolucci E
ModulaTools Various Modula 2 prog. routines. by Jerry Mack
Terrain3d Pseudo-random 3d relief scenery generator, update of "sc", FF87. by Chris Gray, 3d by Howard Hull

Fred Fish Disk 95

Cmd redirects the serial.device or parallel. device output to a file. Capture print jobs, debug or "offline" printng.V4 By C Scheppner SE
CygnusEdDemo Demo of CygnusSoft's CygnusEd editor, a multipefile, multiple feature editor.Includes demo 3.0 of MandFXP. by CygnusSoft Software E
Gomf "Get Outa My Face" makes the Guru go away to allow clean-up & shutdown more cleanly. V 1.0, by Christian Johnsen E
Journal records sequence of mouse & keyboard events, stored in a file for future playback. Good for demos or documtenting bugs. E. by D. Cervone
MergeMem attempts merging of MemList entries of sequentially configured ram boards. Allows allocating a section of memory which spans both boards. V 2, update of FF56. by Carolyn Scheppner SE
PrinterStealer Asimilar to "Cmd", allows diversion of output destined for printer to a file. Binary only, Source avai. from authors. by A. Livshits & J-M Forgeas
Record-Replay similar to "Journal", records and plays back mouse and keyboard events. B only. source avail. from authors, Alex Livshits & J-M Forgeas

Fred Fish Disk 96

AnimPlayer Animation reader and displayer by the combined efforts of Videoscape, Sculpt3D, Silver, Forms- In-Flight, and Animator:Appreticeby M Hashetal.
Chess Amiga port, non-Amiga interface. High playability. V 1.0, S. by J. Stanback, Amiga port by B. Leivian
Hackbench provides source for WB-Likeprog, for experimentation & validation of new interface ideas. Not a WB replacement. by Bill Kinnersley
Label Print labels with arbitray text. V 1.3, Source available from author, M.Hansen
LineDrawer Produces line drawings based on drawing commands stored in a text file. Includes demo that draws an outline map of the USA and state borders. V1.0, SE. by John Olsen
PopUpMenu Example code implementing pop-up menus, reasonably compatible with Intuition menus.SE. by Derek Zahn
Tek4695 Tektronix 4695/4696 printer driver. SE. by P Staub
TimeRam Fast and Chip ram test prog,E by B Takahashi
WarpText Fast text rendering routines, to be linked with application progs.Text display "as fast or faster than 'blitz'". V2.0 update of FF87, S by Bill Kelly

Fred Fish Disk 97

Replaces FF57 for Copywrite problems
CutAndPaste Implementaions of Unix cut and paste commands. by John Weald
Graphit Progran to plot simple functions in 2 or 3 dimensions. by Flynn Fishman
Juggler V1.2 of robot juggler animation. Uses HAM mode and ray tracing. by Eric Graham
MouseReader Shareware program to read text files & view IFF files using only the mouse. by William Betz
Splines Prog to demonstrate curve fitting & ren-dering techniques. by Helene (Lee) Taran
Shm Graphics demo, approximately simulates the motion of two interacting pendulums. Includes S by Chris Ediss

Fred Fish Disk 98

Access 16 color terminal program based on Comm V1.,34. Includes Macro window, custom gadgets, colorized menus, etc. V. Beta 0.18 by Keith Young ,comm by D.J.James. E.

Backup Writes AmigaDos disks as the backup destin-ation. recover files from the backup disk. Requires manual decisions on disk structure. by Alan Kent SE

DCDemo DishCat 2.3, a disk catalog program, demo limited to cataloging 100 files at a time. by Ed Alford, MicroAce Software

HdDriver WD-1002-05 hard disk controller driver. Card capable of maintaining 3 hard disks and 4 floppies, the driver is capable of only one hard disk. by Alan Kent SED

QBase Quick-Base, a "MailBase Management utility", define and maintain a maximum of 200 records per file. by Kevin Harrise E

Thai Thai language quiz program. Speak or type english/Thai sentences from supplied file. by Alan Kent SE

Fred Fish Disk 99

A-Render Version .3 a Ray- Tracing Construction Set for the Amiga Computer by Brian Reed ED

Fred Fish Disk 100

Berserk Must see animation, by Leo Schwab

Conman Console handler replacement, provides line editing and command line histories transparent to application prog uses CON: windows. Shareware V1.0 by W Hawes. E.

WBLander Workbench display hack game, upgrade of "Rocket" on FF85, now with sound effects. By Peter da Silva. E

Fred Fish Disk 101

CirPlane Circular plane generator for VideoScape3D. Generates a clockwise circular polygon with the specified number of vertibles. V1.0 by T Floryan SE

IconAssembler Change Workbench Icons with IFF-brush files by Stefan Lindahl E

Microspell Standalone spelling checker scans text files and reports errors. 1000 common word list, 43,000 word main dictionary with multiple user dictionary support. Interfaces with MicroEMACS 3.9 with an emacs macro to step through the source file, stopping at suspect words and allowing the user to option. V1.0 by Daniel Lawrence, SED

Midi midi library and utility set. Includes Midi monitor, routing utility, status utility, and more. by Bill Barton SED

PsInitrp Postscript Interpreter reads and previews files on screen. by Greg Lee S(assy)E

StartUps Three C startup file replacements for standard Astartup.obj and LStarup.obj. Options include (1) BothStartup.obj, for the WorkBench programs or CLI programs with or without command line parameters. (2) WBStartup.obj, for WorkBench programs or CLI programs that require no command line parameters. (3) CLIStartup.obj for CLI programs that require command line parameters but do not need to be WorkBench runnable. by Bryce Nesbitt SE

Fred Fish Disk 102

Dbug Machine independent macro based C de-bugging package. Update FF41, by F Fish profiling support by Binayak Banerjee SE

Match-stuff Heavy duty text pattern matching stuff. includes simple match text replacement capability. By Pete Goodeve

Sectorama Recover lost or damaged data from floppy or hard disks or repair a damaged volume. by David Joiner E

SiliCon Smart input line interpreter with window for full editing. Upgrade FF50 by P Goodeve., E

Xicon Use icons to call up scripts containing CLI commands. V2.0 upgrade of FF31.by Pete Goodeve E

Fred Fish Disk 103

AvlTrees Library and test prog. implement routines for creating and using trees held in memory.S.

Calc A programmable RPN calculator.

Cref A C cross ref. prog. S.

DosKwik A pair of progs. allows you to save files to one or more floppies for quick loading. Doesn't store Dos format.

IntuiDos A prog. to improve control and handling of the material on all disks in "CLI-area".

MFF-Update A text import util. for MicroFiche Filer (demo on FF 89) and updates to some PD disk library databases.

Pack-It Takes all files the files and dirs. on a disk & packs them into a single file. for modem.

Sol Amiga version of solitaire.

Fred Fish Disk 104

Analyticalc Is a large and powerful spreadsheet prog.

Fred Fish Disk 105

AsmProgs Misc. assembly tools. Includes some S.

BasicProgs LeastSquare solves least square probs ,graphs results,S.

Bison A replacement for unix "yacc" command. S.

Dmouse Another prog in the tradition of display hacks". S.

FlamKey Allows keyboard and mouse inputs to be locked until a password is entered.

GravityWars Game of planets,ships &black holes, v2.0 ,FF84 update.

IPo2C A util. to write a C-lang definition to mimic the intuition pointer.S

Pere-et-Fil Ex. of creating & using reentrant processes. S.

Record Replay Similiar to "Journal" v2.0 update to FF95.

Fred Fish Disk 106

Funckey Shareware function key editor, v1.1update to FF89. Source avail. from author(Anson Mah).

MoreArt A small selection of some Amiga artwork.

QuickFlix An IFF slideshow and cel animation prog,v0.13.

RistiNolla A Finnish game. Also called Go-Moku. v1.0

Fred Fish Disk 107

Csh V2.07 of Matt Dillon's csh like shell.S.

Diff A util.,similiar to other common "diff" programs.S.

ProSuite Provides ex. code of facilities such as FileIO Requester, XText, DoRequest. & tutorial on how to program the Amiga. Book 1.01.S

SVTools Some useful tools. S.

Fred Fish Disk 108

AList Dir listing prog. based on LD4 prg S

DirMaster Disk cataloger. v1.0b, update to FF89. S.

Dots-Perfect Printer Driver for an Epson MX80 printer with upgrade kit installed. S.

MonIDCMP Lets you monitor the IntuiMessages that pass through an IDCMP window. Prints the message class,mouse coordinates,qualifier values. Great for debugging. S.

PrintPop A util. to send common control settings to PRT: S.

Sectorama Utilities to recover lost or damaged data from floppies & hard disks. v1.1, an update toFF102.

Tek Vt100 emulator for a Tektronix 4010/4014. (V2.6) update to FF52. S.

Zoo File archiver, like "arc", v1.24B. update to FF87

Fred Fish Disk 109

Machine A new animation.

SimCPM A CP/M sim,simulates 8080 along with h19 emulationS.

UUpc Hook up your Amiga as a usenet node. S.

Fred Fish Disk 110

A68k A 68000 assembler written in C. S.

Pdc An optimizing C compiler for the 68000 processor. update to FF53. but not based on that code.

Fred Fish Disk 111

AmyLoad A graphical monitor of cpu, blitter, & memory use. Includes two components; load.device,monitors system parameters, & amyload, which is the user interface & display program. by Jeff Kelley SE

AssignDev Assigns multiple names to a given device. modified version of the original released on disk number 79. By Philip Lindsay, mod by Olaf Seibert SE

Gauge Continuously displays memory usage in a vertical bar graph Binary only. By:Peter da Silva

HeliosMouse Another "sunmouse"prog. Automatically activates a window by mouse pointer V 1.1, update to FF94. By:Davide Cervone SE

Labels Alphabetic & numeric ordered cross reference lists of defined system constants. Recommended for debugging purposes only, use the symbolic values in progs! By:Olaf Seibert

Mandel Mandelbrot generator program, with bits & pieces of code from C. Heath & R.J. Mical. By:Olaf Seibert S

PopLife A PopCLI type that plays life all over your screen. Lots of bits & pieces from Tomas Rokicki's blitlab & John Toebes' PopCLI. By:Olaf Seibert S

Fred Fish Disk 112

BeachBirds Beach scene portrayed by sprites & sound 512K machine. By:Jerrold Tunnell B only.

Bully Pushes all open screens around (thus the name "bully"). Show more than one demo at a time By:Mike Meyer S

DropShadow Dropshadow V2.0, use with Bryce Nesbitt's Wavebench demo. B only. By:Jim Mackraz

HagenDemos "RGB" & "Focus". RGB requires one meg, B only. By:Joel Hagen

Viacom Latest version of viacom for use in conjuction with WaveBench demo. B only. By:Leo Schwab & Bryce Nesbitt

WaveBench A neat screen hack, & runs on 512K machines. For more laughs, try in conjunction with Viacom or Ds (Dropshadow). Includes S. By:Bryce Nesbitt

Fred Fish Disk 113

AmiCron Simple Unix "cron" type program,a background task uses a disk-resident table to automatically run certain tasks on a regular basis, at specific times. V 2.3, S. By:Steve Sampson, Amiga port by Rick Schaeffer

Dme V 1.28f of Matt's text editor. A simple WYSIWYG editor for programmers. Not a WYSIWYG word processor Features : arbitrary key mapping, fast scrolling, title-line statistics multiple windows, iconify windows,etc. Update to FF93, S. By:Matt Dillon

DosDev Example DOS device driver in Manx C. Version 1.10, includes S. By:Matt Dillon

M2Amiga Demo of M2Amiga. A fast single pass Modula-2 compiler with editor, linker, a small set of interface & standard libraries. Compiles only small demo programs by limiting codesize & imports. Further development of the ETHZ compiler on FF24. B only. Demos with Source. By:R. Degen, C. Nieder, M. Schaub, J. Straube (AMSoft)

NoIconPos Clears position info of any icons, allows WorkBench to pick a new place for the icon. Useful for disk & drawer icons where Snapshot rewrites the icon & the window information. Modula-2, another demo for M2Amiga By:Markus. Schaub

Fred Fish Disk 114

CDecl English to C (and vice versa) translator for C declarations, a must for anyone except possibly the most hardcore C guru. By:Graham Ross, S

Vt100 V2.7 of vt100 terminal emulator with kermit & xmodem file transfer. Includes a few bug fixes posted to Usenet shortly after the posting of v2.7. Update to FF55. Includes S. By:Dave Wecker

WBLander a special version of the WBLander program from FF100. Ending is unique. Effective use of sound, Includes S. By:Peter da Silva & Karl Lehenbauer

Fred Fish Disk 115

Killer Masterful Video commercial of the Amiga, Beatles music, requires one meg of memory to run. Binary only. By:R. Wilt

Marketroid Another devious sprite oriented demo with lots of "in" jokes. 512K required, includes S. By:Leo Schwab

Fred Fish Disk 116

Movies A ram animation system with three different example animations; Kahnarkas, Rocker, & F-15. Kahnarkas & Rocker run on a 512K Amiga & show oft overscan HAM mode. Includes a animation player program (movie), animation builder programs (dibm, pibm), & a text/graphics display program (vibm). By:Eric Graham & Ken Offer

Fred Fish Disk 117

AMUC_Demo A really neat horizontal scrolling demo that is a 2400 x 200 pixel 32 color IFF picture composed of digiview snapshots of members of the AMiga Users of Calgary, superimposed on a very wide picture of the Calgary Skyline. B only. By:Stephen Vermeulen & Stephen Jeans

ExP_Demo Demo version of Express Paint 1.1., used to create the scrolling demo picture in the AMUC_Demo drawer on thi disk. B only. By:Stephen Vermeulen

Fred Fish Disk 118

Empire Complete rewrite, in Draco, of Peter Langston's Empire . A multiplayer game of exploration, economics, war, etc, can last months.Use local keyboard or modem.V1.0, shareware, & S . By:Chris Gray, original game by Peter Langston

HAMmmm Displays lines whose end points are bouncing around the screen, which is a double buffered HAM screen. The Y positions of the points are continuously copied into an audio waveform and played on all four channels, & the pitch of a just intoned chord is derived from the average X position of these points. J.Forth. Source By:Phil Burk

Stars Based on original code by Leo Schwab, credits longer than actual demo. Runs on 512K Amiga. B only. By:Hobie Orris

WireDemo Demonstrates the Amiga's line drawing speed. Runs on a 512K Amiga. Includes S. By:Matt Dillon

Fred Fish Disk 119

MicroEMACS V3.9e of Daniel Lawrence's variant of Dave Conroy's microemacs. Update to FF93. Also included, for the first time, is extensive documentation in machine readable form. SE. Author: Dave Conroy, Enhanced by Daniel Lawrence

Fred Fish Disk 120

Amoeba Clone of Space Invaders, one of the PDS games for the Amiga. B only BY: LateNight Developments

BackGammon Graphical Backgammon (an undergraduate A.I. course project). Version 1.0, S. By: Robert Pfister

Bankn A complete checkbook system offered by the author as shareware. Version 1.3, binary only. By: Hal Carter

EgyptianRun "road race + hazards" type game. Version 1.1, B only, shareware, source available from author. By: Chris Hames

IconImage Replace an old icon image with a new image, without affecting icontype, drawer data, etc. SE. By: Denis Green

Fred Fish Disk 121

BasicStrip AmigaBASIC prog. helps to convert programs written in other forms of Basic to AmigaBASIC. By: George Trepal

DataPlot Shareware,AmigaBASIC, plotting program. Also includes a least squares curve fit program. By: Dale Holt

Plot Shareware 3-D AmigaBASIC graphing prog. & sample output plots. Source available via author.By: George Trepal

Stairs AmigaBASIC prog. demos a musical illusion based upon perceptual circularity of widely spaced tones whose volumes are defined as a sinusoidal relationship to their frequency. By: Gary Cuba

Uedit V2.3 of this nice shareware editor. With learn mode, command language, menu customization, and other user configurability and customizability features. Binary only, shareware, update toFF60. By: Rick Stiles

WBColors Prog to change Workbench colors for progs that expect to be booted off their distribution disk but are run from a hard disk. SE. Author: Stefan Lindahl

Fred Fish Disk 122

Asteriods Asteroid game. The images and sounds are replaceable by the end user. Anything goes! By: Rico Mariani

If2Pcs Interactive puzzle prog. takes any IFF file with up to 16 colors, and breaks it up into squares to make a puzzle which the user can then piece together. V1.0,S. By: Ali Ozer

Names A shareware program to create and manage mailing lists. Binary only. By: Ernie Nelson

Pr Utility to print listings in different formats. Similar to the Unix "pr" program. Includes source By: Samuel Paolucci

PushOver Board strategy game, AmigaBASIC. Push your pieces onto the board until yfive in a row in any direction.S. By: R.Yost

PuzzlePro Create a puzzle from an IFF picture, which the user can then piece back together again. AmigaBASIC. V1.0, B only, shareware, source available from author. By: Syd Bolton

Fred Fish Disk 123

Arp ARP stands for "AmigaDOS Replacement Project". Arp is an effort led by Charlie Heath of Microsmiths Inc., to replace the current DOS in a compatible fashion, so that current programs will continue to work. Arp also makes whatever improvements are possible, so that current and future programs will work better. Various authors contributed work

Car One of Allen's entries to the Badge Killer Demo Contest. It apparently is an inside joke relating to a well known Amigan's experience with a certain highend graphics hardware manufacturer. Author: Allen Hastings

Fred Fish Disk 124

Icons Some sample animated icons. By: L. Pfost

Tarot AmigaBASIC Nice graphic of tarot cards. Author: L.Pfost

Fred Fish Disk 125

ElGato Animation entry to the BK D Contest. Background music arrangement, requires Sonix to use. By Kevin Sullivans

Fred Fish Disk 126

Colour Manipulate the colors of specific named screens, saving current color sets to data files, loading new color sets from data files, or interactively changing colors. S. By J. Russell

Dance Two programs, "dancing polygons", are entries to the BKD Contest. They aresimilar, but demonstrate the range of colors available on the Amiga. S. By: John Olsen

HBHill Animation entry to the BDK Contest. First known animation using the "Extra Half Brite" mode. By: Kevin Sullivan

Iconify Subroutine creates an icon on the Amiga screen that can be subsequently dragged around, and double-clicked on. You can use this to have your programs "iconify" themselves to temporarily get out of the user's way. With source & demo program. By Leo Schwab

OnlyAmiga Animation entry to BDK Contest. Three balls being juggled by pyramids rotating on their tops. By Iqbal Singh Hans

Suplib Support library needed to rebuild various programs of Matt's from source, including DME, DTERM, etc. S By Matt Dillon

VCheck V1.2 of virus detection prog. from Commodore Amiga Technical Support.. Will test for the presence of a virus in memory, or on specific disks. B only. By: Bill Koester.

Fred Fish Disk 127

Bounce Entry for BDK Contest. Creates little dots that bounce around and multiply. S By: Steve Hansel and Tom Hansel

Nemesis Entry to BDK Contest. It is quite small for what it does, and won fifth place in the contest. B only. By Mark Riley

Ripples Entries to BDK Contest. Unlike most other animations, it shows a fixed object from a moving point of view, instead of vice versa. By: Allen Hastings

Fred Fish Disk 128

Dis 68000 disassembler, written in assembler. S By Greg Lee

DropCloth Place a pattern, a 2 bitplane IFF image or a combination of a pattern and image, into the WorkBench backdrop. Version 2.2, shareware, B By: Eric Lavitsky

LedClock An extremely simple clock program, for interlaced screens only. S By: Ali Ozer

MRBackUp Hard disk backup utility, does a file by file copy on AmigaDOS floppy disks. With an intuition interface & file compression. V1.3, Source. By: Mark Rinfret

Paint Simple screen painting program, written in web. Requires web preprocessing program to rebuild from source. Includes source in web. Author: Greg Lee

PrtDriver A printer driver for the Toshiba "3 in one" printer in its Qume (best) mode. Includes source in C and assembler. By: Rico Mariani

SDBackUp A hard disk backup utility. CLI interface only. Does file compression. V 1.1, binary only. By Steve Drew

Sed A clone of the Unix sed (Stream EDitor) program. Includes source. By: Eric Raymondw

Keys A "hot-keys" program binds keyboard function keys to window manipulation functions (window activation, front to back, moving screens, etc). S By: Davide Cervone

Fred Fish Disk 129

DosKwik A pair of programs which allow you to save files, or a group of files, to one or more floppies for quick loading, does not store files in DOS format, for speed. V2.0, update to FF103. B, Shareware. By: Gary Kemper

MRBackUp A hard disk backup utility, does a file by file copy to standard AmigaDOS floppy disks. Includes intuition interface & file compression. V2.0 (with sources) and 2.1 (binary only, source available from author). Update of FF128. By Mark Rinfret

PaintJet HP PaintJet printer driver from HP sources.

Patch Two independent ports of Unix utility "patch", which applies context diffs to text file to automatically update them. Patch V 1.3 was ported to the Amiga by Rick Coupland and patchV2.0 was ported by Johan Widen. S By: Larry Wall

Fred Fish Disk 130

DirMaster Shareware disk cataloger, V1.1, update of FF108, new features and enhancements. B only. By Greg Peters

Evo Human evolution toy/tutorial with source.By S. Bonner

Hp RPN calculator prog, supports calculations with binary, octal, decimal, hex, float, and complex numbers. Includes 32 registers for storing data & transcendental functions. V1.0, S By: Steve Bonner

Mach "mouse accelerator" prog. with hotkeys, features of sun mouse, clicktofront, and popcli, a title bar clock with a bbs online charge accumulator, etc.V1.6a, S By: Brian Moats

PatEdit A pattern editor for creating patterns to input to the Amiga SetAtPt macro call. Call sets the area fill pattern for the area filling graphics (RectFill, AreaDraw, etc). Includes source. By Don Hyde

QMan Mandelbrot generator written partially in assem. for speed. Includes source. By Steve Bonner

Fred Fish Disk 131

Dfc Copies disks like Maurader, but multitasks. Replaces diskcopy and format (smaller than either). Intuition interface. S By: Tomas Rokicki

HyperBase Shareware database management system. V1.6, Binary only, source available from authors. FF58 update. By: Michael MacKenzie, Marc Mengel, & Craig Norborg

Life A new version of Tomas's ancient Life game, with a new macro language for setting up patterns, good examples, S By: Tomas Rokicki

Mackie A Popcli replacement that draws pretty lines on the screen in blanking mode. Includes source. Author: Software Distillery; enhancements by Tomas Rokicki

Mg1b A version of Mg1b with an ARexx port and other improvements by Tomas Rokicki. Define macros & bind them to function keys in startup file. Includes source. Author: Various; enhancements by Rokicki

WFrags Another version of Frags, Pops up a little window that updates occasionally. Good for developers to monitor what progs. are doing to memory. S By: Tomas Rokicki

Fred Fish Disk 132

Berserk Animation, a "must see" for every Amiga user, and ranks with "Juggler" as a premier demo for the Amiga. The difference between this distribution, and FF100, this one includes "source". use it as an example for creating animations. Fred Fish felt it was appropriate to have at least one animation that was available at the "source code" level. Author: Leo Schwab

Fred Fish Disk 133

Conman Shareware replacement for the standard console handler, provides line editing and command line histories completely transparent to any application program that uses CON: windows. V1.1, binary only, updateoFF100. New features include additional editing keys, fast search keys, undo key, clear history command, and more. Author: William Hawes

Crc Two programs useful for generating 16-bit CRC listings of the contents of disks, and verifying that a given disk's files still compute to the same CRC's as listed. V1.0, binary only. By: Don Kindred

CrcLists Complete CRC check files for FF1-128 using the Crc program included on this disk. These were made directly from Fred's master disks. Author: Fred Fish

Overscan Patches the Intuition library so that sizable windows with MaxHeight of 200 (400 in interlace) and screens with Height of 200 (400 in interlace) will take advantage of the PAL overscan capability of Intuition V1.2. Useful only for European users who wish to run software written for the US market, without modifying the applications, but still using the additional space. S By Ari Freund

Fred Fish Disk 134

BoingThrows 50 frame HAM animation done with Sculpt-3D, and DigiPaint. The animation took about 325 hours of runtime to generate. By Marvin Landis

Browser Workbench tool, using text-only windows, makes all files in the system accessible for executing, copying, moving, renaming, deleting, etc. Billed as a "programmers workbench". V1.2, binary only. BY Peter da Silva

Dme V1.29 of Matt's text editor. Simple WYSIWYG editor designed for programmers. Arbitrary key mapping, fast scrolling, title-line statistics multiple windows, & ability to iconify windows. FF113 update,S. By Matt Dillon

For PDS orders, please use form on page 112

Find Utility searches for files that satisfy a given boolean expression of attributes, starting from a root pathname and searching recursively down through the hierarchy of the file system. Like the Unix find program. V1.0, includes source. By Rodney Lewis

Library Demo version of a shareware program that stores textual information without regard to structure or content, and allows complicated searching for specific patterns. , B only. By: Bill Brownson

SmartIcon Shareware Intuition objects iconifier. V1.0 is limited to iconifying windows, adds a new "iconify gadget" to each window, when clicked, iconifies the window into an icon in the ram: disk. B only, source available from author. By: Gauthier Groult

Fred Fish Disk 135

TeXF A selection of 78 TeX fonts, with a conversion program to convert them to Amiga fonts. 22 different fonts at various sizes, ranging from 15 pixels high to more than 150 pixels. Conversion program can also be used with the fonts distributed with AmigaTeX, yielding an additional 1000+ fonts for use withother Amiga programs. V2.5, binary only. By: Ali Ozer

Fred Fish Disk 136

AsmToolBox Assembler "toolbox" created to make interfacing between assembler programs and AmigaDOS easy. With source. By Warren Ring

Bison A replacement for unix "yacc" command. From the GNU (GNU is Not Unix) effort. Port of the latest GNU version, by William Loftus, with the goal of preserving all of bison's current features. Includes source & test pro. "calc". By: Bob Corbett and Richâ Stallman.

Iff2Pcs Interactive puzzle progr. takes any IFF file containing up to 16 colors, and breaks it into squares to make a puzzle the user can then piece back together again. V1.1, update of FF122, includes source. By Ali Ozer

Paste Version of the Unix paste utility. Paste concatenates corresponding lines of the specified files into a single output line (horizontal or parallel merging) or concatenates them into alternate lines (vertical or serial merging). S. By: David Ihnat

YaBoingII Game prog. demonstrating hardware sprite usage, including collision detection. Update of FF36. S. By: Ali Ozer, based on original by Leo Schwab

Zoo File archiver, like "arc" in concept, but different in implementation and user interface details. Includes features that "arc" lacks (such as file/path names up to 255 characters in length). V 1.71, update of FF108. B. By: Rahul Dhesi, Amiga port by Brian Waters

Fred Fish Disk 137

Ct Program to display images from a CT scanner, along with several interesting sample images of scans of real people, including a skull, brain, heart, and spine. Each image is 256 by 256 pixels in 2048 gray scale. The display software, though it has a primitive user interface, is quite powerful, including functions like convolutions, averaging, laplacians, unsharp masking, edge detection, gradients, etc. Binary only. Author: Jonathan Harman

JeansIcons Miscellaneous cute Icons created for AMUC's monthly newsletter disk. Submitted by Stephen Vermeulen. Author: Steve Jeans

Muncho A cute little program which plays a digitized sound sample when you insert or remove a disk from your drive. If you don't like the sounds, you can replace them with your own. Binary only. By: Andrew Werth

Sit Update to the Set Icon Type prog. on FF107. V1.10, includes source. Author: Stephen Vermeulen

VGad A new gadget editor that takes two pictures of the window and its gadgets, one being the normal gadget state and the other being the fully selected state, then merges the data and converts to C source code. V1.0, binary only. Author: Stephen Vermeulen

VirusX A boot sector virus check program that runs in the background and automatically checks all inserted disks for a nonstandard boot sector. Such disks can optionally have their boot sector rewritten to remove the virus. Includes source. Author: Steve Tibbett

VLabel Program to print fancy customized disk labels. Combines an IFF picture and up to 50 lines of text (which may be placed arbitrarily in any font or point size) then print the result. The IFF picture can be virtually any size (up to 1008 by 1000). It will also print labels from a batch file produced by SuperBase. V1.20, binary only. By: Stephen Vermeulen

Fred Fish Disk 138

AmigaLine A series of various technical notes for Amiga programmers. By: Byrce Nesbitt

Diff Uses the same algorithm as the Unix diff program and also produces context diffs, suitable for use with patch. Binary only. By: Unknown (Decus C diff?)

Foreach A simple but useful program that expands a wild card file specification and then invokes the specified command once per expanded filename, with the expanded filename as the command argument. Includes source. Author: Jonas Flygare

MacFont A conversion tool to convert Mac fonts to Amiga fonts. Binary only. By: John O'Neill and Rico Mariani

ModulaTools Various useful routines for those using in Modula on the Amiga. Update to FF94, S . By: Jerry Mack

Vt100 Two new versions of Dave's vt100 terminal emulator. One version, based on vt100 2.6, has been enhanced by John Barshinger to include an iconify feature, add full 132 column support using overscan, and other features (binary only). The second version is release 2.8 of the main-stream version of vt100, as enhanced and supported by Tony Sumrall. S. By: Dave Wecker

Fred Fish Disk 139

AmiCron An enhanced and debugged version of AmiCron 2.3 from FF113. Includes source. By: Steve Sampson, Rich Schaeffer, Christian Balzer

ListScanner A nice little utility to display all the Exec lists. Simular to Xplor utility FF73. Includes source in assembler. By: Heiko Rath

ProCalc Simulates HP-11C programmable calculator. Both English & German versions. Shareware, B only. By: Gotz Muller

RemLib Removes a specified library (if currently unused) or displays some info on all available libraries. Source in assembler. By: Heiko Rath

TurboBackup A fast mass floppy disk duplicator with enforced verify mode to prevent errors. V1.0, binary only. By: Steffen Stempel and Martin Kopp

WArranger Sends a window, identified by its name, to the front or back, without selecting it. Useful with AmiCron. Works on all screens. Includes source in assembler. By: Heiko Rath

WheelChairSim A wheelchair simulator developed as a project for the Technical Resource Centre and the Albert Children's Hospital, to allow the matching of a wheelchair joystick to a child's handicap and allow the child to practice using the chair in a safe (simulated) environment. Binary only. Author: Unknown, submitted by Dr. Mike Smith

Fred Fish Disk 140

SBProlog Volume 1 of the 2 volume Stony Brook Prolog (SBP) distribution, V2.3.2. This volume contains the executables and libraries. Volume 2, on FF141, contains the C and Prolog Source. By: Logic Programming Group at SUNY, Stony Brook Amiga port by David Roch & Scott Evernden

Fred Fish Disk 141

SBProlog Volume 2 of the 2 volume Stony Brook Prolog (SBP) distribution, version 2.3.2. Volume 2 contains the C and Prolog source code. Volume 1, on FF140. By: Logic Programming Group at SUNY, Stony Brook Amiga port by David Roch and Scott Evernden

SmallC An Amiga port of the Small-C compiler, written by Ron Cain and published in Dr. Dobb's Journal, in about 1980. Small-C is a rather small subset of the full "C" language. It is capable of compiling itself, and other small, useful programs. Requires an assembler and linker to complete the package and produce working executables. Source and binary. By: Ron Cain. Amiga port by Willi Kusche.

Fred Fish Disk 142

Diff Program uses same algorithm as Unix diff prog. and produces context diffs, suitable for use with patch.Same as FF138, but now includes the missing files (including source code). Author: Unknown (Decus C diff)

FracGen Generates fractal pictures from "seeds" you create. Unlike any of the other "fractal generators", it can be used to load and display previously created fractal pics, modify existing fractals, or create your own fractals. V1.1, B. By:Doug Houck

SciSubr Scientific Subroutine Package from DECUS, ported to the Amiga to run with Absoft Fortran. A valuable resource of mathematical and statistical source code for those doing Fortran work on the Amiga. Author: Unknown; ported to the Amiga by Glenn Everhart

Fred Fish Disk 143

Rim RIM-5 (Relational Information Manager), a full relational DBMS suitable for VERY large databases using B-Tree data storage, crude (by today's standards) user interface, but full source code is provided. RIM runs on a wide variety of systems, small and large, and produce compatible databases. Includes a built in HELP database and a programming language. Full Fortran source code and documentation included. Author: Various, Amiga port by Glenn Everhart

Fred Fish Disk 144

AnalytiCalc V22-3D of Glenn Everhart's large and powerful spreadsheet program, update to FF104. Extra features "to have some pretentions of acting as an 'integrated system'. A virtual memory system supporting up to18000 columns and 18000 rows, multiple equations per cell, an outlining system, built-in cell annotation, and datafile access from any cell(s) of the sheet, plus an array of functions not present in most commercial spreadsheets. Source and documentation in arc'd form.

Fred Fish Disk 145

Csh Modification of csh like shell to provide file name completion and argument execution. Requires ARP 1.1. Binary only, but includes diffs for the reference 2.07 source base. Author: Matt Dillon; enhancements by Johan Widen

DMouse Versatile screen blanker, mouse blanker, auto window activator, mouse accelerator, popcli style programmable command key, pop window to front, push window to back, etc, widget. Very useful program! . V1.06, includes source. Author: Matt Dillon

Net Link protoco provides essentially an unlimited number of reliable connections between processes on two machines, where each can be either an Amiga or a Unix (BSD4.3) machine. Works on the Amiga with any EXEC device that looks like the serial.device. Works on UNIX with tty and socket devices. Achieves better than 95% average throughput on file transfers. V1.20, includes sources for both the Amiga and Unix versions. Author: Matt Dillon

Tab Tablature writing program, with intruments for a banjo and string guitar. Binary only. Author: Jeff deRienzo

TinyProlog VT-PROLOG is a simple prolog interpreter provided with full source code to encourage experimentation with the PROLOG language and implementations. Version 1.1, includes source. Author: Bill and Bev Thompson

Fred Fish Disk 146

Blanker2 A screen blanking program that turns the screen black after 90 seconds of keyboard and mouse inactivity. V1.27.88, includes source. Author: Joe Hitchens

C-Light A demo copy of a commercial ray tracing program, identical to commercial version but limited to ten objects per scene. Binary only. Author: Ronald Peterson

CrcLists Complete CRC check files for FF129-141 and FF143-145 of the library, using the crc program from FF133. Made directly from Fred's master library. FF142 omitted due to a problem with the crc program. Author: Fred Fish

DmeMacros A set of DME macros which utilize templates to turn DME into a language-sensitive editor for C, Pascal, Modula-2, and Fortran. By Jerry Mack

MemoPad A shareware intuition-based memo reminder program. Nicely done. V1.1, binary only. Author: Michael Griebling

Fred Fish Disk 147

MicroGNUEmacs MicroGNUEmacs(MG 2b) contains many additions and enhancements since the original works by Dave Conroy (credit belongs to all contributors and Beta testers. Note: Amiga specific source code files and the document files have been archived. An executable copy of the PDS archive program "Zoo" is in the "c" directory

Fred Fish Disk 148

EFJ "Escape from Jovi" A machine-code game featuring hi-res scrolling, large playfield, disk-based Hi-Score list, stereo sound, and multiple levels. Use a joystick in port 2 to control the ship. B, shareware ($8). By: Oliver Wagner

Fme Nicely done map editor for the Fire-Power (tm) game. Features interlaced hi-res with intuition interface. See the "Readme.fnf" file for information on making a bootable disk. Includes source. Author: Gregory MacKay

HandyIcons Adds a menustrip to the WorkBench window that allows you to run selected Workbench Tools by menu selection. Can be set up to provide custom environments. Current version supports only WorkBench Tools and not Projects. Binary by: Alan Rubright

Scrambler A simple program that will encode/decode a text file into illegible gibberish, which resembles executable code, to evade prying eyes. Version 0.01, Binary only. Author: Foster Hall

Fred Fish Disk 149

AnimalSounds A sample of digitized animal sounds along with a simple sound player. Authors: The Trumor Company, Inc. Sound Player by Don Pitts

DX-VoiceSorter Written to be used with Jack Deckard's VoiceFiler program. (Disk 82). It allows for the sorting of a number of voicefiles stored using that program into a new voicefile of voices made up from various files. Includes source. Author: David Bouckley

Keep A nice little utility program with an intuition interface for BBS and network junkies who download messages in one large file and then read them off-line. Using only the mouse, you can drive through such files a message at a time, examine each at your leisure and tag those you wish to keep. Version 1.2, binary only, but source available with donation to author. Author: Tim Grantham

Less Like Unix "more", only better, with forward and backward scrolling, searching and positioning by percent of file and line number, etc. Now lets you also print the current file. Very useful! This is Amiga version 1.3, an update to the version on disk number 92. Includes source. Author: Mark Nudelman, Amiga port by Bob Leivan

Scheme "Scheme is a statically scoped and properly tail-recursive dialect of the Lisp programming language invented by Guy Lewis Steele Jr. and Gerald Jay Sussman." Binary only. Amiga port by Ed Puckett

Fred Fish Disk 150

AirFoil An update to the Airfoil generator on disk #71. Generates airfoil models as well as their corresponding streamline and pressure distributions. Includes source. Authors: Russell Leighton Addendum by David Foster

DC10 An AmigaBasic DC-10 instrument flight simulator. Appears to be quite in-depth with flight-planning and take-off options along with an extensive documentation file. Requires rebuilding on a separate disk and was successfully done so by following the author's instructions in the ReadMe_First file. Author: Jan Arkesteijn

ExecLib A working example of how to build and use user-defined disk-resident libraries. Of special interest to developers working with Lattice C. Author: Alex Livshits

Iconizer A utility program that saves your current mouse pointer to a small icon. You can restore the pointer just by double-clicking on its icon. Allows for building a whole library of pointers and to use them whenever you want. Binary only. Author: Alex Livshits

Pilot An implementation of the PILOT language for the Amiga, including a demo done for the National Park Service. PILOT is a limited use language for use in educational and computer based instruction programs. Binary only with Beta test kit available from authors. Author: Terry LaGrone

StealMemBoot A small utility designed to be a direct replacement for NoFastMem kind of programs. It modifies the boot block of a disk, so when you boot with it, all memory allocations will return only CHIP memory. Author: Alex Livshits

Fred Fish Disk 151

GlobeDemo A graphics demo which displays very smooth transitions of the rotating earth. Features a pop-up menu. Includes source. Author: Bob Corwin

Icons Yet another potpourri of interesting icons to choose from if you need one for your own program. Author: Dave Tumock

Pcopy A small intuition-based disk copier similiar to the resident "DiskCopy" except with write-verify and other user-selectable options. Useful for making multiple copies with reliable data. Requires two disk drives. Includes source. Author: Dirk Reisig

SCT A CLI-based utility (SetColorTable) for displaying and/or setting a screen's colors. You can save the colors of a screen to be restored later, or copy one screen's colors to another. Includes source. Author: aklevin

SlideShow Very nicely done slide-show program written in assembly language. Features forward/backward presentation and creative screen wipes. Currently works only with IFF lores pictures. Executable only along with some new IFF pictures to have come my way. Shareware ($16). Authors: Mike McKittrick and Sheldon Templeton

Surveyor A little utility that opens a window on the current screen and displays information about the pointer. Allows for absolute or relative measurement between two points on the screen. Very handy for precise positioning of icons and such. Includes source. Author: Dirk Reisig

Fred Fish Disk 152

Blk A requester making tool employing various recursive algorithms including a recursive parser. It takes input text files and converts them to C-source for including as requester declarations. Includes source. Author: Stuart Ferguson

RunBack A variant of Rob Peck's RunBackGround program from disk number 73. Allows you to start a new CLI program and run it in the background, then closes the new CLI. This version automatically searches the command-search-path to find the program. Includes source. By Daniel Barrett

UUCP This is a version of uucp (Unix to Unix Copy Program) for the Amiga, along with some miscellaneous support utilities like cron, mail, and compress. Includes source. Author: Various, submitted by William Loftus

Fred Fish Disk 153

Dme Version 1.30 of Matt's text editor. Dme is a simple WYSIWYG editor designed for programmers. It is not a WYSIWYG word processor in the traditional sense. Features include arbitrary key mapping, fast scrolling, title-line statistics multiple windows, and ability to iconify windows. Update to version on disk number 134, includes source. Author: Matt Dillon

HP11 Emulates an HP11C caiculator including the program mode. Features an ON/OFF button that turns the calculator into an icon that will sit and wait until you need it again. Documentation on the features is scarce, perhaps some industrious HP owner could write a small tutorial for the benefit of those that don't own an HP calculator. Binary only. Author: David Gay

HPMam A program to manipulate settings and fonts on HP LaserJet+ printers and compatibles. Includes an intuition interface and some sample picture files. Version 1.0, binary only, shareware. Author: Steve Robb

Synthemania An interesting, very small (and very persistent!) musical piece. If you plan on stopping it without using three fingers, you better read the document file first! Binary only. Author: Holger Lubitz

Fred Fish Disk 154

Ada An Ada Syntax checker for the amiga. Includes lex and yacc source. Author: Herman Fischer; updates by William Loftus

AssemblyDemos A interesting group of assembly language demos for your visual and aural pleasure. Binary only. Author: Foster Hall

DiskLib Two utilities for those people who like to split up PD disks into disks of different categories. Includes source. Author: Wilson Snyder

Guardian Another virus diagnosing and vaccination program. Recognizes any non-standard bootblock. Includes a small utility program to permanently place the program on a copy of your kickstart disk in place of the seldom (if ever!) used Debug() function. Binary only. Author: Leonardo Fei

PrintSpool A print-spooling program. Very useful for printing files in the background. Many command-line options. Version 1.0.0, Includes source. Author: François Gagnon

Utilities A group of four little utility programs:

- UnDelete - Undelete a file from floppy (DF0:) to any device you request, checks for a disk in the drive and allows you to abort cleanly with a CTRL 'C'.
- WhereIs - Looks for a file and or directory defaults to the current device
- CAL - Clone of the Unix CAL command, dates from the year 1 to 9999.
- DClock Simple title bar clock/memory gauge with pop to front.

VirusX An update to the virus-detecting program of the same name on disk number 137. This version also checks for the Byte-Bandit strain. Version 1.21, Includes source. Author: Steve Tibbett

Virus_Alert! Yet another anti-virus program with a twist. Once installed a message is displayed just after a warm or cold boot notifying the the user that the disk and memory are virus-free, and forcing a mouse-button press before continuing. Anything writing to the boot-block thereafter will destroy the message and a normal virus- infected boot (???) will take-place. Versions 1.01 and 2.01, Binary only. Author: Foster Hall

Wicon A "Window Iconifier". Allows you to turn your windows into small icons which can be later recalled. Currently installed with MacWin to give your windows a "rubber-banding" effect. Version 1.14, Includes source. Author: Steven Sweeting Introducing the Amiga

To Be Continued.....

In Conclusion

To the best of our knowledge, the materials in this library are freely distributable. This means they were either publicly posted and placed in the public domain by their authors, or they have restrictions published in their files to which we have adhered. If you become aware of any violation of the authors' wishes, please contact us by mail.

IMPORTANT NOTICE!

This list is compiled and published as a service to the Commodore Amiga community for informational purposes only. Its use is restricted to non-commercial groups only! Any duplication for commercial purposes is strictly forbidden. As a part of Amazing Computing™, this list is inherently copyrighted. Any infringement on this proprietary copyright without expressed written permission of the publishers will incur the full force of legal actions.

Any non-commercial Amiga user group wishing to duplicate this list should contact:

PiM Publications, Inc.
P.O.Box 869
Fall River, MA 02722

PiM Publications Inc. is extremely interested in helping any Amiga user groups in non-commercial support for the Amiga.

•AC•

For PDS orders, please use form on page 112

Amaze Me

Please use this order form when subscribing to Amazing Computing™, ordering Back issues, or ordering Amiga™ Public Domain Software

Name_____
Street_____
City _____ **St.**___ **Zip** _____
Amount Enclosed _____

Please circle the appropriate item: **New Subscription** **Renewal**

Please start my subscription to Amazing Computing™ with the next available issue or renew my current subscription.
I have enclosed $24.00 for 12 issues in the U.S. ($36.00 Canada and Mexico, $44.00 foreign surface).
All funds must be in U.S. Currency on a U.S. Bank

Back Issues:
$5.00 each US, $6.00 each Canada and Mexico, $7.00 each Foreign Surface.

Please circle your Back issue choices below:

Vol1.1	Vol1.2	Vol1.3	Vol1.4	Vol1.5	Vol1.6	Vol1.7	Vol1.8	Vol1.9	Vol2.1	Vol2.2	Vol2.3	Vol2.4	Vol2.5
Vol2.6	Vol2.7	Vol2.8	Vol2.9	Vol2.10	Vol2.11	Vol2.12	Vol3.1	Vol3.2	Vol3.3	Vol3.4	Vol3.5	Vol3.6	Vol3.7
Vol3.8	Vol3.9												

Public Domain Software:
$6.00 each for subscribers (yes, even the new ones!)
$7.00 each for non subscribers
(three disk minimum on all foreign orders)

Please circle your Public Domain Software choices below:

Amazing on Disk:

A#1...Source & Listings V3.9

Amicus:

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13
A14	A15	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26

Fred Fish:

FF1	FF2	FF3	FF4	FF5	FF6	FF7	FF8	FF9	FF10	FF11	FF12	FF13
FF14	FF15	FF16	FF17	FF18	FF19	FF20	FF21	FF22	FF23	FF24	FF25	FF26
FF27	FF28	FF29	FF30	FF31	FF32	FF33	FF34	FF35	FF36	FF37	FF38	FF39
FF40	FF41	FF42	FF43	FF44	FF45	FF46	FF47	FF48	FF49	FF50	FF51	FF52
FF53	FF54	FF55	FF56	FFNA	FF58	FF59	FF60	FF61	FF62	FF63	FF64	FF65
FF66	FF67	FF68	FF69	FF70	FF71	FF72	FF73	FF74	FF75	FF76	FF77	FF78
FF79	FFNA	FF81	FF82	FF83	FF84	FF85	FF86	FF87	FFNA	FF89	FF90	FF91
FF92	FF93	FF94	FF95	FF96	FF97	FF98	FF99	FF100	FF101	FF102	FF103	FF104
FF105	FF106	FF107	FF108	FF109	FF110	FF111	FF112	FF113	FF114	FF115	FF116	FF117
FF118	FF119	FF120	FF121	FF122	FF123	FF124	FF125	FF126	FF127	FF128	FF129	FF130
FF131	FF132	FF133	FF134	FF135	FF136	FF137	FF138	FF139	FF140	FF141	FF142	FF143
FF144	FF145	FF146	FF147	FF148	FF149	FF150	FF151	FF152	FF153	FF154		

(NA denotes disks removed from the collection)

Please complete this form and mail with check or money order to:

PiM Publications, Inc.
P.O.Box 869
Fall River, MA 02722

Please allow 4 to 6 weeks for delivery

Amazing Computing Reader Service Card

Want to know more about Amazing Computing advertisers and their products? Here's your chance! Fill out the card below (for instructions, see page 96) and drop it in the mail (US only). We will process your request promptly and forward it to the advertiser.

Quick, easy, and efficient—just like all services of Amazing Computing: Your Original Amiga Monthly Resource.

FREE INFORMATION

Amazing Computing Reader Service Card

Name_____

Street_____

City _____ St.____ Zip _____

Telephone_____

October card Valid until 12/31/88

Please see page 96 for instructions.

101	102	103	104	105	201	202	203	204	205
106	107	108	109	110	206	207	208	209	210
111	112	113	114	115	211	212	213	214	215
116	117	118	119	120	216	217	218	219	220
121	122	123	124	125	221	222	223	224	225
126	127	128	129	130	226	227	228	229	230
131	132	133	134	135	231	232	233	234	235
136	137	138	139	140	236	237	238	239	240
141	142	143	144	145	241	242	243	244	245
146	147	148	149	150	246	247	248	249	250
151	152	153	154	155	251	252	253	254	255
156	157	158	159	160	256	257	258	259	260
161	162	163	164	165	261	262	263	264	265
166	167	168	169	170	266	267	268	269	270
171	172	173	174	175	271	272	273	274	275
176	177	178	179	180	276	277	278	279	280
181	182	183	184	185	281	282	283	284	285
186	187	188	189	190	286	287	288	289	290
191	192	193	194	195	291	292	293	294	295
196	197	198	199	200	296	297	298	299	300

A. Do you own an Amiga?
- O 1. Amiga 1000
- O 2. Amiga 500
- O 3. Amiga 2000
- O 4. Soon
- O 5. Not Yet
- O 6. Just Looking

B. What Amiga hardware product do you plan to buy next?
- O 1. Amiga 500
- O 2. Amiga 2000
- O 3. Memory Expansion
- O 4. Hard Drive
- O 5. IBM Emulators (sidecar or Bridgeboard)
- O 6. Printer
- O 7. Modem
- O 8. Music Tool
- O 9. Video Product
- O 10. Other

C. What Amiga software product do you plan to buy next?
- O 1. 'C' Language
- O 2. Forth Language
- O 3. Modula-2 Language
- O 4. Assembly Language
- O 5. BASIC Language
- O 6. Entertainment
- O 7. Telecommunications
- O 8. Spreadsheet
- O 9. Database
- O 10. Financial
- O 11. Video
- O 12. Graphics
- O 13. Music
- O 14. Other

D. Where do you buy your Amiga products?
- O 1. Local Amiga Dealer
- O 2. Discount Department Store
- O 3. Manufacturer
- O 4. Mail Order

E. Which type of articles do you like to see in Amazing Computing?
- O 1. 'C' Language
- O 2. Forth Language
- O 3. Modula-2 Language
- O 4. Assembly Language
- O 5. BASIC Language
- O 6. Game Reviews
- O 7. Business Reviews
- O 8. Hardware Product Reviews
- O 9. Software Product Reviews
- O 9. Programming How To's
- O 10. Business How To's
- O 11. Video Articles
- O 12. Graphics Articles
- O 13. Music Articles
- O 14. Hardware How To's
- O 15. PDS Updates
- O 16. Interviews
- O 17. Other

F. Which articles would you like to see more of in Amazing Computing?
- O 1. 'C' Language
- O 2. Forth Language
- O 3. Modula-2 Language
- O 4. Assembly Language
- O 5. BASIC Language
- O 6. Game Reviews
- O 7. Business Reviews
- O 8. Hardware Product Reviews
- O 9. Software Product Reviews
- O 9. Programming How To's
- O 10. Business How To's
- O 11. Video Articles
- O 12. Graphics Articles
- O 13. Music Articles
- O 14. Hardware How To's
- O 15. PDS Updates
- O 16. Interviews
- O 17. Other

G Are you a subscriber to Amazing Computing?
- O 1. Yes
- O 2. No

101

Amazing Computing Reader Service Card

Name_____

Street_____

City _____ St.____ Zip _____

Telephone_____

October card Valid until 12/31/88

Please see page 96 for instructions.

101	102	103	104	105	201	202	203	204	205
106	107	108	109	110	206	207	208	209	210
111	112	113	114	115	211	212	213	214	215
116	117	118	119	120	216	217	218	219	220
121	122	123	124	125	221	222	223	224	225
126	127	128	129	130	226	227	228	229	230
131	132	133	134	135	231	232	233	234	235
136	137	138	139	140	236	237	238	239	240
141	142	143	144	145	241	242	243	244	245
146	147	148	149	150	246	247	248	249	250
151	152	153	154	155	251	252	253	254	255
156	157	158	159	160	256	257	258	259	260
161	162	163	164	165	261	262	263	264	265
166	167	168	169	170	266	267	268	269	270
171	172	173	174	175	271	272	273	274	275
176	177	178	179	180	276	277	278	279	280
181	182	183	184	185	281	282	283	284	285
186	187	188	189	190	286	287	288	289	290
191	192	193	194	195	291	292	293	294	295
196	197	198	199	200	296	297	298	299	300

A. Do you own an Amiga?
- O 1. Amiga 1000
- O 2. Amiga 500
- O 3. Amiga 2000
- O 4. Soon
- O 5. Not Yet
- O 6. Just Looking

B. What Amiga hardware product do you plan to buy next?
- O 1. Amiga 500
- O 2. Amiga 2000
- O 3. Memory Expansion
- O 4. Hard Drive
- O 5. IBM Emulators (sidecar or Bridgeboard)
- O 6. Printer
- O 7. Modem
- O 8. Music Tool
- O 9. Video Product
- O 10. Other

C. What Amiga software product do you plan to buy next?
- O 1. 'C' Language
- O 2. Forth Language
- O 3. Modula-2 Language
- O 4. Assembly Language
- O 5. BASIC Language
- O 6. Entertainment
- O 7. Telecommunications
- O 8. Spreadsheet
- O 9. Database
- O 10. Financial
- O 11. Video
- O 12. Graphics
- O 13. Music
- O 14. Other

D. Where do you buy your Amiga products?
- O 1. Local Amiga Dealer
- O 2. Discount Department Store
- O 3. Manufacturer
- O 4. Mail Order

E. Which type of articles do you like to see in Amazing Computing?
- O 1. 'C' Language
- O 2. Forth Language
- O 3. Modula-2 Language
- O 4. Assembly Language
- O 5. BASIC Language
- O 6. Game Reviews
- O 7. Business Reviews
- O 8. Hardware Product Reviews
- O 9. Software Product Reviews
- O 9. Programming How To's
- O 10. Business How To's
- O 11. Video Articles
- O 12. Graphics Articles
- O 13. Music Articles
- O 14. Hardware How To's
- O 15. PDS Updates
- O 16. Interviews
- O 17. Other

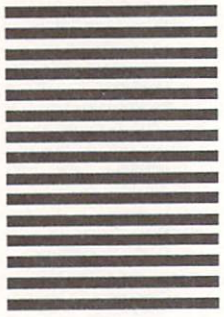
F. Which articles would you like to see more of in Amazing Computing?
- O 1. 'C' Language
- O 2. Forth Language
- O 3. Modula-2 Language
- O 4. Assembly Language
- O 5. BASIC Language
- O 6. Game Reviews
- O 7. Business Reviews
- O 8. Hardware Product Reviews
- O 9. Software Product Reviews
- O 9. Programming How To's
- O 10. Business How To's
- O 11. Video Articles
- O 12. Graphics Articles
- O 13. Music Articles
- O 14. Hardware How To's
- O 15. PDS Updates
- O 16. Interviews
- O 17. Other

G Are you a subscriber to Amazing Computing?
- O 1. Yes
- O 2. No

102

FREE INFORMATION

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 36 FALL RIVER, MA

POSTAGE WILL BE PAID BY ADDRESSEE

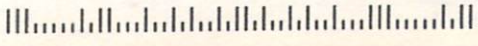
Amazing COMPUTING
P.O. Box 869
Fall River, MA 02722-0869

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 36 FALL RIVER, MA

POSTAGE WILL BE PAID BY ADDRESSEE

Amazing COMPUTING
P.O. Box 869
Fall River, MA 02722-0869



Do It NOW!!!

Subscribe TODAY and save over 49% on Amazing Computing!

Amazing Computing is available to United States subscribers at $24.00 for 12 issues. That's a savings of $23.40 off the newsstand price. Canada and Mexico, $36.00. Foreign Surface to all other countries, $44.00
(Air mail and first class rates available upon request.)

Enter my 12 issue subscription today!

Name_____

Street_____

City _____ St.____ Zip _____

Amount Enclosed _____

Sorry, we cannot accept credit cards or requests for billing at this time.
All funds must be in U.S. Currency drawn on a U.S. Bank.
Please allow four to six weeks for processing.

Clip and $AVE !!

(over 49% of the newsstand price!)

Save Time and Money!

Subscribe TODAY and save over 49% on Amazing Computing!

Amazing Computing is available to United States subscribers at $24.00 for 12 issues. That's a savings of $23.40 off the newsstand price. Canada and Mexico, $36.00. Foreign Surface to all other countries, $44.00.
(Air mail and first class rates available upon request.)

Enter my 12 issue subscription today!

Name_____

Street_____

City _____ St.____ Zip _____

Amount Enclosed _____

Sorry, we cannot accept credit cards or requests for billing at this time.
All funds must be in U.S. Currency drawn on a U.S. Bank.
Please allow four to six weeks for processing.

Special Subscription Bonus

See the special offer on the inside rear cover of this issue

Special Subscription Bonus

See the special offer on the inside rear cover of this issue

Please Mail to:

Amazing COMPUTING

P.O. Box 869
Fall River, MA 02722-0869

Place this order form in an envelope with your Check or Money Order.

Clip and $AVE !!

(over 49% of the newsstand price!)

Please Mail to:

Amazing COMPUTING

P.O. Box 869
Fall River, MA 02722-0869

Place this order form in an envelope with your Check or Money Order.

Don't Miss An Issue!

Subscribe TODAY and save over 49% on Amazing Computing!

Amazing Computing is available to United States subscribers at $24.00 for 12 issues. That's a savings of $23.40 off the newsstand price. Canada and Mexico, $36.00. Foreign Surface to all other countries, $44.00. (Air mail and first class rates available upon request.)

Enter my 12 issue subscription today!

Name_____

Street_____

City _____ St.____ Zip _____

Amount Enclosed _____

Sorry, we cannot accept credit cards or requests for billing at this time.
All funds must be in U.S. Currency drawn on a U.S. Bank.
Please allow four to six weeks for processing.

Please Mail to:

Amazing COMPUTING

P.O. Box 869
Fall River, MA 02722-0869

Place this order form in an envelope with your Check or Money Order.

I Will NOT Miss Out!!!

I will subscribe TODAY and save over 49%on Amazing Computing, plus receive my Amazing Product Guide: Winter '89.

I can't pass up an offer like this! Enclosed is my check for $24 for a one-year subscription to Amazing Computing. Canada and Mexico, $36.00. Foreign Surface to all other countries, $44.00.
(Air mail and first class rates available upon request.)

Enter my 12 issue subscription today!

Name_____

Street_____

City _____ St.____ Zip _____

Amount Enclosed _____

Sorry, we cannot accept credit cards or requests for billing at this time.
All funds must be in U.S. Currency drawn on a U.S. Bank.
Please allow four to six weeks for processing. We are not responsible for late enteries.

Please Mail to:

Amazing COMPUTING

P.O. Box 869
Fall River, MA 02722-0869

Place this order form in an envelope with your Check or Money Order.

Don't Miss Out!

ALL AC subscribers will receive the Amazing Product Guide: Winter '89 absolutely FREE!

The Amazing Product Guide: Winter '89 will be a complete descriptive listing of *all* products available for the Amiga. If you have appreciated the theme-specific product guide installments in past few issues of AC, just imagine bigger and better chunks of information. The *Amazing Product Guide: Winter '89* will feature longer, more detailed descriptions and even more products!

The Guide will be available on newsstands in December at the standard cover price of $3.95. All our subscribers, Australians, Canadians, Americans, Europeans,and even our rock-sitting stegosaurus dragon, will receive the Guide along with their December AC for no additional charge! The $24 that usually gets you a year full of great Amiga information through Amazing Computing, can now bring you that same year of information-packed AC *and* the *Amazing Product Guide: Winter '89*.

Don't Miss Out!

The *Amazing Product Guide: Winter '89* will be a must for all Amiga users (fire-breathing or not), but your Amazing Dealer will sellout quickly! Guarantee your *Amazing Product Guide: Winter '89* by mailing your subscription request today!

Order Early!

Your subscription request MUST be received in time for our mailing. NO exceptions! Mail TODAY!

New!
Version 3.0
includes overscan, half-brite support and more!

DIGI VIEW

DELIVERS ULTIMATE GRAPHICS POWER

Bring the world into your Amiga with Digi-View, the 4096 color video digitizer. In seconds you can capture any photograph or object your video camera can see in full color and with clarity never before available on a home computer. Digi-View's advanced features include:

- Dithering routines give up to 100,000 apparent colors on screen
- NewTek's exclusive Enhanced Hold-and-Modify mode allows for exceptionally detailed images
- Digitize images in any number of colors from 2 to 4096
- Print, animate, transmit, store, or manipulate images with available IFF compatible programs
- Digitize in all Amiga resolution modes (320x200, 320x400, 640x200, 640x400)

"Digi-View sets new standards for graphics hardware"-*InfoWorld*

Digi-View is available now at your local Amiga dealer or call:
1-800-843-8934

ONLY $199.95

NEWTEK
INCORPORATED

All photos actual unretouched Digi-View pictures shot directly off the 1080 Amiga monitor.